Implementation of Enterprise Applications based on Service Oriented Architecture

Amar Nath

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India
June 2014

Implementation of Enterprise Applications based on Service Oriented Architecture

Dissertation submitted in partial fulfillment of the requirements for the degree of

Master of Technology

in

Computer Science and Engineering

(Specialization: Software Engineering)

by

Amar Nath

(Roll No.- 212CS3125)

under the supervision of

Prof. S. K. Rath

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, 769008, India

June 2014

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India.

Certificate

This is to certify that the work in the thesis entitled ***"Implementation of Enterprise Applications based on Service Oriented Architecture"*** by ***Amar Nath*** is a record of an original research work carried out by him under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Master of Technology with the specialization of Software Engineering in the department of Computer Science and Engineering, National Institute of Technology Rourkela. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Place: Rourkela
Date: June 2, 2014

(Prof. Santanu Ku. Rath)
Professor, CSE Department
NIT Rourkela, Odisha

Acknowledgment

First of all, I bow my head in reverence to the Almighty, most benevolent and merciful, the creator of the universe, for providing me with this opportunity to work with the intelligentsia and enabling me to reach far beyond my own, restricted ambit of thought and action

It is my great pleasure to express my sincere and profound thanks to my learned supervisor Prof. Santanu Kumar Rath Chair-Department of Computer Science and Engineering, NIT Rourkela, who has given me enormous support, guidance and encouragement. He has provided me guidance and many useful suggestions during the study. His observations and comments helped me to establish the overall direction to the research and to move forward with in depth investigation. He has helped me greatly and been a source of knowledge. I am very much indebted to my supervisor, for his continuous encouragement and support. He is always ready to help with a smile. I am also thankful to all the professors of the department for their support.

Secondly, I would like thank my wife Archana and daughter Akshita for their love, without which this work would not have been possible. I thank to all my friends especially Dr. Virendra Kumar, Mr. Hari Kishan Saini, Mr. Kaka Singh, Mr. Hariom, Mr. Sharvan Kumar, Mr. Ankit Kumar. Further I would like to thank all my lab-mates i.e. Mr. Y. Suresh, Mr.Shashank, Mr. Abinash, Mr. Ashish Dwedi, Mr. Mukesh Kumar, Mrs. Sumana, Mrs. Jyoti Shivhare, Mrs. Prerna Kanojia and Mr. Lov Kumar for their encouragement and moral support which kept me going through this work. Their help can never be penned with words.

Finally, I would like to dedicate this thesis to my family i.e. my Parents, loving wife Archana and daughter Akshita, brother Aadesh, and Niece Janu for their love, patience, and understanding.

Amar Nath

Roll-212cs3125

Abbreviations

ANN	Artificial Neural Network
ATM	Automatic Teller Machine
BPEL	Business Process Execution Language
EA	Enterprise Architecture
ESB	Enterprise Service Bus
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
PNML	Petri net Markup Language
REST	Representational State Transfer
RPC	Remote Procedure Call
RST	Rough Set Theory
SaaS	Software as a Service
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
UDDI	Universal Description Discovery and Integration
WSDL	Web Service Description Language
XML	Extended Markup Language
XSD	XML Schema Definition

Abstract

In current scenario, It has become important to integrate business and its Information system to achieve their competitiveness. Today a large number of projects fail in achieving business to IT integration. Distributed systems which are large in nature are difficult to develop due to their complexity. The Service Oriented Architecture (SOA) has been identified as the most suitable solution which facilitates to development of distributed systems. SOA helps in developing such distributed system as it supports modular design, inter-interoperability, application integration and software reuse. Enterprises systems which are large in nature need to focus on characteristics i.e. scalability and availability that enhance interoperability and integration of elements. An enterprise is an organization of resources i.e. people, computers, and machines, which performs a process. Enterprise Architecture (EA) is the discipline control resources of the enterprise. EA tasked to ensure Business-IT alignment. Enterprise SOA can be defined as a set of business-aligned IT-services that collectively address an organization's business processes (work-flow) and goals. Services can be orchestrated in many ways to support enterprise business process. Services and Processes are guided by the business architecture and it can be traced back to the business outcomes.

Service Oriented Architecture (SOA) help in bridging the gap between business and IT by well defined, business-oriented services developed by using design principles, frameworks, pattern and methods. The objectives of EA and SOA are quite similar. However, whereas EA is a framework that covers all dimensions of enterprise IT architecture, SOA provides an architectural strategy that uses the concept of a Service as the underpinning business-IT alignment entity. SOA concept can be implemented by using open standards, i.e. SOAP, WSDL and UDDI. These all standards are based XML due to which SOA supports interoperability between services operating on different platforms and between applications implemented in different programming languages.

In this thesis, Stock Price Prediction and Bank Automatic Teller Machine

(ATM) applications are designed and implemented by using the SOA as a basis for designing, implementing, deploying, invoking and managing their services. In stock price data collection application SOA is being used for collecting stock price and soft-computing techniques for efficient prediction. In this study also the main characteristics of SOA-based ATM components and services is identified, designed and implemented. To ensure the correctness of services orchestration, a formal verification is done using a Petri net model of BPEL process.

***Keywords*:** Artificial Neural Network (ANN), Automatic Teller Machine, Java Business Integration (JBI), PNML, Rough Set Theory (RST), Service Oriented Architecture (SOA), Stock Price Prediction.

Contents

Certificate ii

Acknowledgment iii

Abbreviation iv

Abstract v

List of Figures x

List of Tables xii

1 Introduction 1

1.1 Introduction 1

1.1.1 Service-Oriented Architecture 1

1.1.2 SOA: A Blueprint for Change 2

1.1.3 Key Ingredients of SOA 2

1.1.4 Basic SOA Paradigm / (Find, Bind, Execute) Paradigm . . 2

1.1.5 Principles, Basic Drivers and Components of SOA: 5

1.1.6 Realization or Implementation of SOA Concept 9

1.2 Why to Adopt SOA? 10

1.2.1 Demerits & Risks of SOA Implementations: 11

1.3 Enterprise Architecture in the Context of Services Orientation . . . 12

1.4 Motivation 12

1.5 Objective 13

1.6 Thesis Organization 13

2 Stock Price Prediction using Service-Oriented Architecture and Soft Computing techniques. 14

2.1 Introduction 15

2.2 Applications of SOA 15
2.3 Techniques used for Stock Price Prediction 16
2.3.1 Methodology of Rough-Set Theory 16
2.3.2 Methodology of ANN technique 16
2.3.3 Testing Phase : 19
2.4 Stock Price Prediction Implementation 19
2.4.1 Collection and Transformation of Stock Price Data 21
2.4.2 Dimension Reduction Technique 22
2.4.3 Normalization 22
2.4.4 Prediction using ANN Technique 23
2.4.5 Prediction 23
2.5 Performance Criteria 24
2.6 Results and Accuracy 25
2.7 Summary 26

3 Design and Implementation of Bank ATM based on Service Oriented Architecture and its Validation with Petri net model 27

3.1 Introduction 28
3.1.1 Proposed SOA based Architecture of ATM system 29
3.2 Literature Survey 30
3.3 Fundamentals of Techniques Used 31
3.3.1 Service Oriented Architecture 31
3.3.2 Petri Nets 32
3.4 Description of Proposed Work 35
3.5 Experiment Details 37
3.5.1 Case study-Withdraw Scenario based on SOA 37
3.6 Development of BPEL Process for Withdraw Service of ATM . . . 39
3.6.1 Business Process Execution Language(BPEL) 39
3.7 Transforming BPEL Process to Petri net 41
3.8 Results and Analysis 41
3.8.1 Average Number of Tokens in a place 42
3.8.2 Token Probability Density 42

3.8.3 Throughput of Timed Transition 44
3.8.4 Sojourn Time for Tangible States 44
3.8.5 Reachability Graph of *WithdrawService's* BPEL 44
3.8.6 State Space Analysis 45
3.9 Summary 46
4 Conclusion and Future Work **47**
Bibliography **49**
Dissemination of Work **53**
5 Annexure **54**
5.1 Source Code of Withdraw Service. 55
Annexure **54**

List of Figures

1.1 Ingredients of SOA 3
1.2 Basic SOA Architecture 3
1.3 Sample WSDL for WithdrawInputWSDL 4
1.4 REST based Data Stored and Accessed 5
1.5 Web Service with their methods 7
1.6 Enterprise Service Bus 8
1.7 Realization / Implementation SOA 9
1.8 Architecture Ontology 12

2.1 A Typical Single Layer FFNN 17
2.2 BPEL and CASA for Stock Data Collection Application 20
2.3 Request and Response SOAP Request Message for Apple stock quote 21
2.4 Predicted vs Actual Stock Price M/S Apple 25
2.5 Predicted vs Actual Stock Price M/S SBI 26

3.1 ATM Architecture 28
3.2 Work-flow Adopted in This Study 30
3.3 Proposed SOA based ATM Architecture 30
3.4 Stochastic Petri net and its corresponding Reachability Graph . . . 33
3.5 ATM Components with their Services 36
3.6 Activity Diagram of ATM Cash Withdraw Scenario 38
3.7 List of Services Identified, Developed and Deployed on Different Components 39
3.8 SOAP Request and Response Message for Withdraw Service 40
3.9 bpelWithdraw 40

3.10 BPEL to PNML Conversion Command 41
3.11 Petri net Model of Withdraw Service 42
3.12 Reachability graph of Withdraw Service 45
3.13 State Space Analysis 46

5.1 Request and Response of Final *CheckBalance* Service 54
5.2 Request and Response of Final *Withdraw* Service 54

List of Tables

1.1 Basic Drivers and Components Relationship 9

2.1 Stock Sample of M/S Apple, USA 21
2.2 Testing and Training Details 23
2.3 Parameters considered for Training 23
2.4 Result and Accuracy 25

3.1 Average Number of Token on a Place 43
3.2 Token Probability Density 43
3.3 Throughput of Timed Transition 44
3.4 Sojourn Time for Tangible States 44

Chapter 1

Introduction

1.1 Introduction

1.1.1 Service-Oriented Architecture

Service-Oriented Architecture (SOA) basically, can be defined as collection of services which communicate to each other independent of technology and platform in which they are deployed. SOA is turning as a design philosophy, which is independent of any vendor, product, technology or industry trend, and bridging the gap between business and IT system trough a set of business-aligned IT service using a set of design principles, pattern and techniques.

SOA also can be defined as architectural style used to designing distributed systems based on loosely coupled, abstracted services. SOA will probably become the de-facto standard for designing and developing the enterprise applications. It tries to bring a solution for the problem of interoperability between heterogeneous systems. SOA is also can be considered as reference architecture that can be applied in organizations that are part of a chain or network, and organizations that have to deal with a lot of regulatory changes, or fast-changing markets. It makes it possible to change parts of the IT landscape, and processes, without affecting other parts of the system. SOA can help enterprise streamline processes to do their business in more efficient manner, and they can tackle changing needs and competition, by enabling the software as a service concept.

Definitions of SOA:

- *SOA is a form of technology architecture that adheres to the principles of service-orientation. When realized through the Web services technology platform, SOA establishes the potential to support and promote these principles throughout the business process and automation domains of an enterprise-* ***Thomas Erl, Chief Architect, XMLTC Consulting Inc*** [1].

- "*Service Oriented architecture is the policies, practices, and frameworks, that enable application functionality to be provided and requested as a sets of services published at a granularity relevant to service requester, which are abstracted away from the implementation using a single, standards based form of interface -* ***Everware-CBDI***" [2].

1.1.2 SOA: A Blueprint for Change

The need to respond to business agility with flexible IT solutions has led many enterprises to adopt SOA, is an IT-framework that integrate individual business services, to implement sophisticated business applications and processes. Agnostic-services are independent and can be used with any application, and run at any computing platforms. SOA allows you to design solutions as assemblies of agnostic services in which the assembly description of services is a managed [3].

1.1.3 Key Ingredients of SOA

Key ingredients of SOA are shown in Figure 1.1. The SOA ingredients are represented by three dimensions i.e. Technology, People, and Process. These ingredients describe what SOA is all about [4].

1.1.4 Basic SOA Paradigm / (Find, Bind, Execute) Paradigm

The Find, Bind, and Execute" paradigm is the basic architecture of SOA that has three major components i.e. service-consumer, service-provider and service registry. Service-registry has service-description, and it gives the consumer a contract

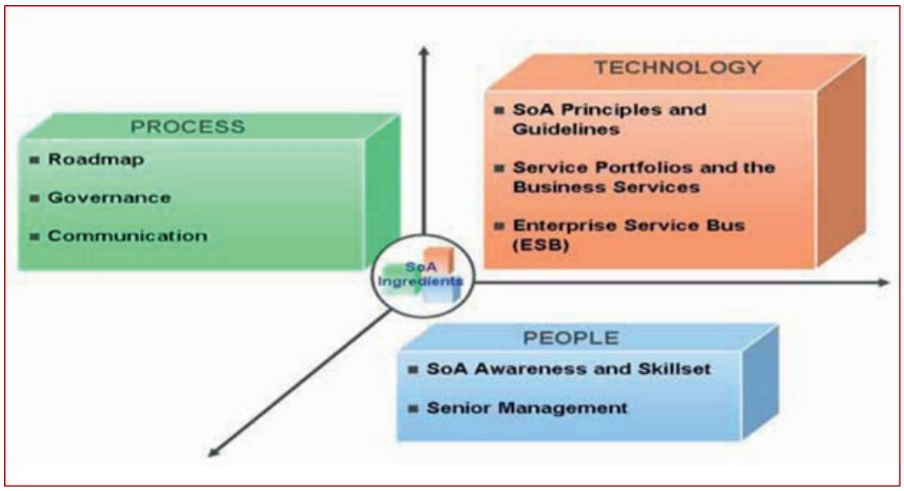

Figure 1.1: Ingredients of SOA

and an endpoint address for the service. Service-provider develop the service and publish it to service-registry as is shown in Figure 1.2.

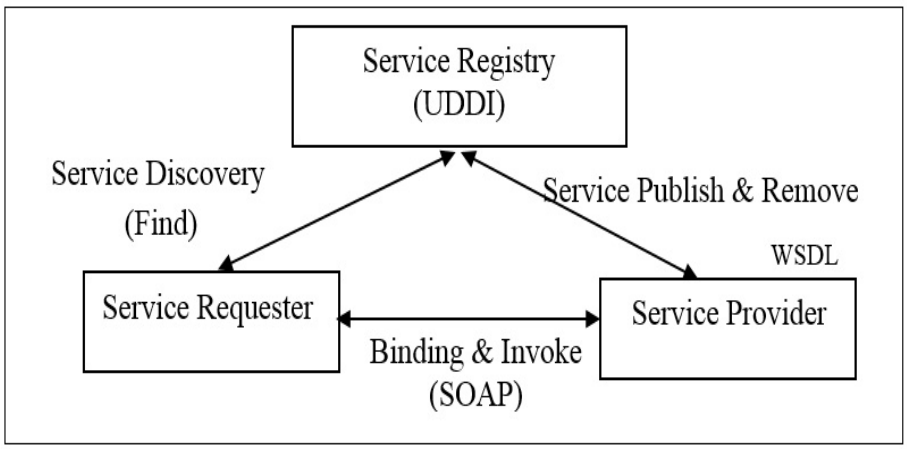

Figure 1.2: Basic SOA Architecture

As per the SOA basic architecture process has the following steps:

1. The process start from service provider who creates any service by using Web Service Descriptive Language (WSDL).

   Web Service Descriptive Language: WSDL XML based grammar for describing communications regarding web services in a structured and standardized way. A sample WSDL for *WithdrawInputWSDL* Service is shown in Figure 1.3.

Figure 1.3: Sample WSDL for WithdrawInputWSDL

2. To make this service available for consumer, service-provider publish this service to service-registry using WSDL. Universal Description, Discovery and Integration (UDDI) protocol based service-registry is a network-based directory that has the details of available services and it also provides the end point address of that service. UDDI, first specification proposed in 2000 by IBM, Ariba and Microsoft. The UDDI Registry is a global, public, online directory that helps enterprise's businesses to describe their services, discover other services, that are provided by other service-providers.

3. Service consumer discover the desired service from UDDI service-registry and if it is found then UDDI registry return the end point address of that service. Now there is direct binding between service consumer and service provider. The communication takes place by some means of protocols i.e. SOAP and REST.

   Simple Object Access Protocol (SOAP): SOAP communication protocol for exchanging of information in a decentralized, distributed environment. SOAP can communicate in heterogeneous environment as it is based on XML.

   Representational State Transfer (REST): REST is an another protocol for realizing the SOA that can be considered as architecture style for

designing distributed networked applications. REST uses a simple HTTP protocol to make calls between hydrogenous machines. It is Resource based rather than action and Representation of resource state between client and server. It relies on a stateless, client-server, cache-able communications protocol and in virtually all cases, the HTTP protocol is used. Six constraints of REST are Interface, Stateless, Client- Server, Client- Server, Cache-able and Layered System.

A REST based example is shown in Figure 1.4.

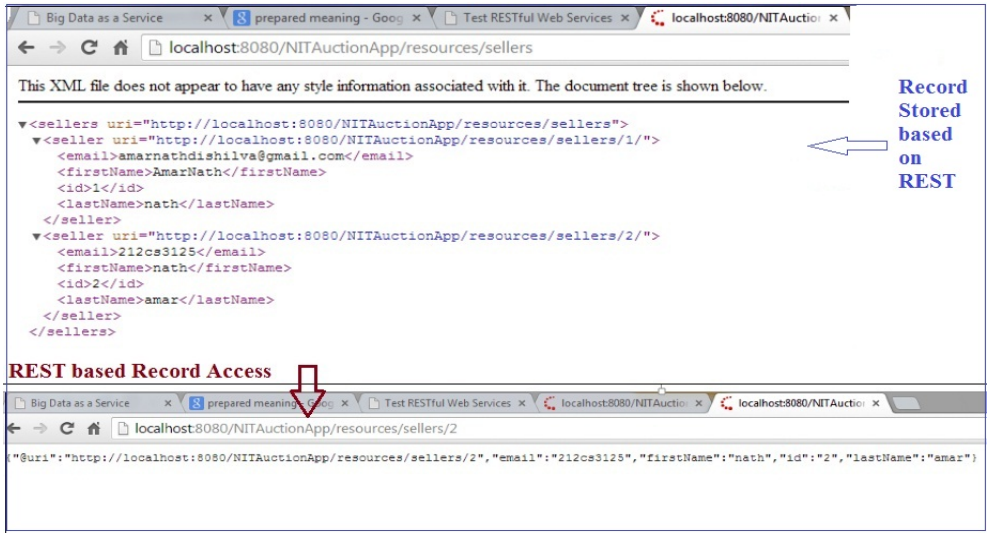

Figure 1.4: REST based Data Stored and Accessed

1.1.5 Principles, Basic Drivers and Components of SOA:

Service-orientation design principles are defined for developing the solution logic of services within service-oriented architectures as there is a lack of industry standards relating to the exact composition of a service-oriented architecture.

1.1.5.1 Principles of SOA are as follows:

- **Standardized Service Contract:** Service-contract is the principle which describes each service and defines the conditions of information exchange between consumer and provider of services.
- **Service Loose Coupling:** Services are designed in a way that they are

independent to other services and one service can be replaced by another without any difficulty. The services are independent in respect technology, platform, process and vendor.

- **Service Abstraction:** The visible part of service to outside world is exposed via the service contract.
- **Service Re-usability:** Services are developed and deployed once and can be positioned as reusable enterprise resources if required.
- **Service Autonomy:** The services are self-descriptive and self-governing. They contain sufficient logic to run and control to it self independently.
- **Service Statelessness:** Services don't keep the information about state of service. It results in reduction of the resources consumed by a service because it is engaged in actual state data management only.
- **Service Discover-ability:** Services should allow their descriptions to discovered by service consumer so that they could invoke them for their use.
- **Service Composability:** Services-composing allows logic to be represented at different levels of granularity to provide more meaning to smaller logic of service. It promotes re-usability and the creation of abstractions.
- **Interoperability:** It provides the ability to invoke and run services among heterogeneous environment.

1.1.5.2 Basic drivers of SOA:

- **Re-usability:** Reusability is the basic driver of SOA, as it is a design principle used for creating the services that have the potential to be reused across the enterprise
- **Integration:** SOA helps for coordinating the work, connect disparate systems and enable manageability from the IT and business perspective. SOA based enterprise is vital to ensure seamless integration between systems, services, and applications.

- **Agility:** Agility is the ability to change business work-flow which has been evolved due to the change in requirement, in a quick manner and SOA based solutions enable it.

1.1.5.3 Basic Components of SOA are:

- **Services:** Service is basic building block of SOA, and it can be define as a repeatable business task and is logical encapsulation of self contained business functionality i.e. check customer credit. Service should be define themselves for example if some one asked to them

  "Hey service, what can you do?"

  Their reply to this query would be -

  "These are my methods, members, properties.. You can call me by these methods and get your output".

  The SBIPINvalidation service which takes the Card Number and PIN of type string as shown is Figure 1.5.

Figure 1.5: Web Service with their methods

- **Enterprise Service Bus (ESB):** Enterprise service bus is the heart of SOA based solution and without it, it is difficult to have a truly scalable and adaptable SOA solution. ESB is an integration platform that connects the various service layers and provides consumers with access to services. An ESB is an architectural pattern that acts as the mediator/ Broker between

the service provider and service consumer. ESB form the communication infrastructure for service to interact with one another across locations and is responsible for interaction.

ESB provides some key features need by any SOA i.e. Service Binding, Messaging, Protocol Switching, Security, Load Balancing, Validation, Monitoring, Routing and Transformation Mechanism.

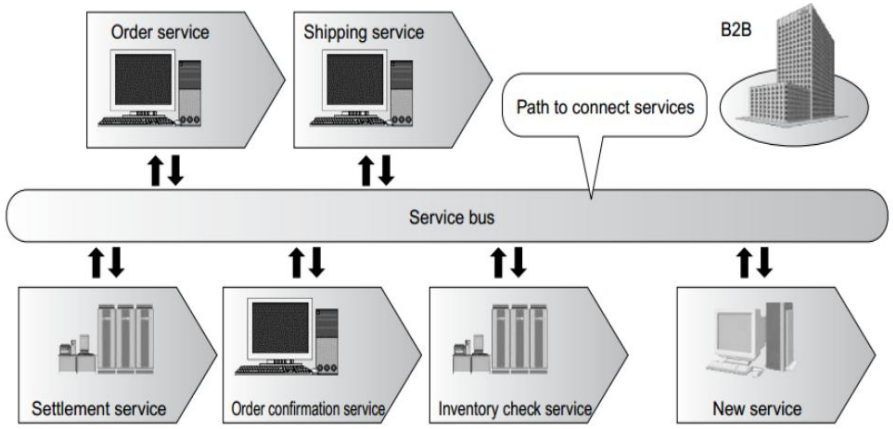

Figure 1.6: Enterprise Service Bus

- **Orchestration:** Orchestration of services brings agility to the business enterprise and is achieved by Business Process Executive Language (BPEL). A business process work-flow is defined by invoking the appropriate service at each stage in the business process. To execute a business process, the work-flow defined is Orchestration .

1.1.5.4 The relationship between basic drivers and components:

The relationship between basic drivers and component is shown in Table 1.1. The re-usability can be achieved by services in the organization. ESB is responsible for integration of applications which are deployed on similar or hydrogenous environment. The agility of business enterprise is tackle by using the concept of orchestration of different services.

Table 1.1: Basic Drivers and Components Relationship

Drivers	Components
Re-usability	Service
Integration	ESB
Agility	Orchestration

1.1.6 Realization or Implementation of SOA Concept

1.1.6.1 Technology:

Enterprise moving towards SOA to re-design their IT system to meet the future requirements. SOA is no longer buzz word, it is slowly moving into the mainstream. SOA can be implemented by using web services, REST, CORBA, J2EE etc as shown in Figure 1.7

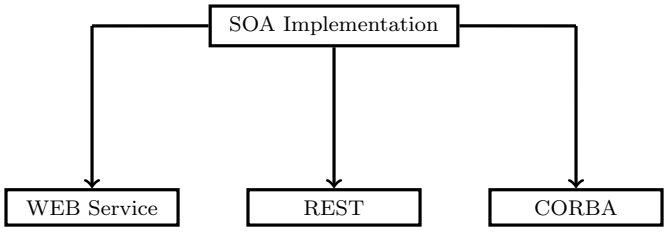

Figure 1.7: Realization / Implementation SOA

1.1.6.2 Web Services

Web services make it possible that systems on any platform can communicate, can be used in a range of application integration scenarios. A web service is used to implement and is becoming de-facto standard of services in SOA and uses the WSDL, UDDI and SOAP protocol.

1.1.6.3 Tools:

- **OpenESB:** OpenESB is Net-Beans based ESB tool for building Integration and SOA applications. Designed and developed by Sun Microsystems, OpenESB is improved and maintained by our OpenESB-community. Relying on JBI (Java Business Integration), OpenESB proposes a unique development process which promotes migration to real service oriented develop-

ment and organization. The information about SOA tool is available at `http://www.open-esb.net/`. It can be downloaded from the same site.

- **Red Hat JBoss Developer Studio:** Jboss is an open source tool by Red Hat. Jboss supports the development of SOA-based applications. The plug-in for SOA is available at `https://devstudio.jboss.com/updates/5.0/soa-tooling/`

- **Oracle Jdeveloper Studio:** Oracle provides a free integrated environment that facilitates Java based development of SOA application. Related information about tools and plug-ins is available at `http://www.oracle.com/technetwork/developer-tools/jdev/overview/index.html`

- **IBM SOA Foundation (RSA/RAD):**

  IBM is one of the biggest player in the SOA market. SOA based application can be developed by IBM RSA and IBM RAD. Information is available at `http://www-03.ibm.com/software/products/en/ratisoftarch`.

In this study, to realizing (Creating web service, BPEL) the SOA **OpenESB** tool is used.

1.2 Why to Adopt SOA?

There are number of reasons to go for SOA-based implementation of enterprise architecture i.e. increased competitions , enhancement of business capabilities, reduced development cost etc.

As per Gartner's report, SOA-based implementation has following improvements

1. Reduce cost (63%)
2. Improve business agility (60%)
3. Support changes in business models (50%)

4. Streamline the IT environment (59%)

5. Support new business processes (57%)

6. Reduce time to solution (53%)

1.2.1 Demerits & Risks of SOA Implementations:

1.2.1.1 Demerits

As each coin has two faces, adoption of SOA also has some disadvantages too. The demerits of SOA are as follows:

- One considerable disadvantage is the complexity of typical service-oriented architecture. Getting to grips with the bigger picture of SOA certainly takes longer than it might otherwise. Indeed, the same goes for creating this big picture in the first place. If done right, laying the foundations for a service-oriented system will most likely take longer than it would if a more monolithic approach were taken.

- Requires Security and high availability.

- Application that may run on standalone machine may avoid SOA; for instance, a word processing application that does not require request and response based calls.

1.2.1.2 Risks in SOA Implementations

Risks may arise from reasons rooted in the SOA solution components if it is not implemented correctly. If the services are not identified, analyzed and detailed carefully during the analysis phase, they becomes very difficult to make any change at the implementation phase. So analysis phase and consideration of candidate service and operation are very important which are identified at this stage.

1.3 Enterprise Architecture in the Context of Services Orientation

Enterprise Architecture (EA): EA is a discipline for business processes and IT infrastructure specially for holistically leading enterprise to reflecting the integration and standardization requirements of the enterprise's operating model. EA is applied to guide decision making toward the evolution of the future state architecture. SOA is particular building technique to implement the enterprise architecture more efficiently. A SOA with EA ontology is shown in Figure 1.8.

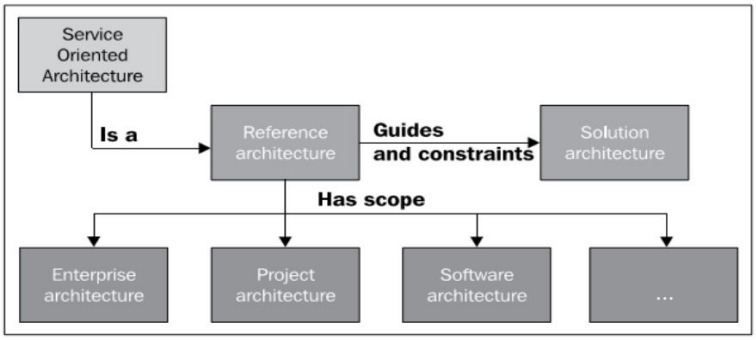

Figure 1.8: Architecture Ontology

1.4 Motivation

Today we are living in the age of global market where business activities are carried out and control from all over the world. Globalization of business, agile markets, high competitive pressures, and unexpected evolving customer need are placing tremendous pressure on IT to deliver greater flexibility with high speed. To overcome to these challenges, enterprises are moving towards adopting service-oriented architecture (SOA) to make their system scalable and responsive to agility. SOA facilitates the development of distributed enterprise applications based on web services with capability of easily integration and reuse. So, this was the motivational reason to carried out this research.

1.5 Objective

The main objective of the present study is to develop architecture for enterprise applications based on SOA and their implementation by using existing tools and techniques. The orchestration of the services is done with the help of BPEL. BPEL is an XML-based language which helps in composition of services to gain better and desirable value of the services. BPEL is based on textual syntax language and need to verified it's correctness by formal method. In this study, PIPE2 tool is used for formalizing the correctness properties about the reliability of the business process design.

1.6 Thesis Organization

The rest of the thesis is organized as follows.

Chapter-2: Describe the application of SOA with Soft-Computing to predict the stock price of a company. First stock data is collect being collected by using the concept of service oriented architecture and then one of the soft-computing technique i.e. Rough Set Theory (RST) has been applied to curtail the data attributes. Artificial Neural Network (ANN) has been used to predict the stock price of a company.

Chapter-3: Propose the design and implementation of Automatic Teller Machine(ATM) based on SOA. In this process first SOA based architecture has been scratched and then services are identified, developed and implemented using open source tool named OpenESB. The orchestration process of ATM for check balance and withdraw service is developed and further this process has been formally verified by Petri net based modeling language using PIPE analysis tool.

Chapter-4 Conclusion and Future Work

Chapter 2

Stock Price Prediction using Service-Oriented Architecture and Soft Computing techniques.

In this chapter, the concept of Service-Oriented Architecture(SOA) is applied for collecting the stock price data of a company and then soft computing techniques, for example, Rough-Set Theory(RST), Artificial Neural Networks(ANN), are used for stock price prediction.

Stock Price Prediction of any company is a challenging task as its data are continuous, highly intensive, multi-dimensional and dynamic in nature. Most of the approaches available on prediction in the literature till date, are based on numerical modeling. This study makes an attempt to extend the concept of numerical modeling and presents an approach for stock price prediction of a given company by incorporating the data collection methodology of Service Oriented Architecture (SOA) and then, applying soft computing techniques for prediction purpose. In this study, the concept of SOA is being applied for collecting stock price data of a particular company using Java Business Integration (JBI) framework. The features of SOA ensure reusability, flexibility, adaptability and integrity of the data collection process. The attributes of obtained dataset are curtailed using one of the soft computing techniques i.e., Rough Set Theory (RST). RST technique works by finding the relevant attributes and eliminating the redundant attributes those are not essential. The residual attributes are used to predict stock price based on Artificial Neural Network (ANN) technique.

2.1 Introduction

Stock market prediction is an important task for active markets and challenging task under financial time series analysis, because of the uncertainties involved in the movement of the market. Various reasons influence the stock market including, political events, general global economic conditions, and companys internal management etc. There are a number of commonly used volatility prediction techniques such as multiple time-series, auto-regression, auto-regressive moving average (ARMA) etc., which are applied for prediction [5]. These techniques are dominated by time series analysis. As the complexity of prediction increases, some other techniques of stock price prediction such as data mining and machine intelligence have been applied and observed to be more helpful. Inductive learning based approaches such as k-nearest neighbor and neural network for stock prediction have been developed using historical price data, which have drastically improved the performance of prediction [1]. The SOA helps the data collection by making the process automatic. SOA provides an interoperable and integrative technical architecture and is implemented by technology of web services for collecting information. After collection of the right kind of data, soft computing techniques are applied to stock price prediction with an objective to achieve accuracy.

2.2 Applications of SOA

SOA is essentially, a collection of services that communicates with each other, either by simple data passing or involving two or more services coordinating some activity [6]. In this study SOA have been applied for collecting the Stock price data by using existing service developed by `www.webservicex.net`.

The key elements of SOA architecture are as follows:

- **Service Provider:** The service-provider provides services. In this study `www.webservicex.net` has been considered as a service provider.
- **Service Requester:** Service requester is a client for a particular service.

In this study, a program termed as *stockQuoteTrace* is treated as service requester that invokes current stock price information service.

- **Service Registry:** Service registry is a mediator between service requester and service provider. The service published to service-registry by service producer is used by the service consumer. Service-consumer search for desired service in service-registry.

JBI based implementation of SOA uses Simple Object Access Protocol(SOAP) protocol. SOAP allows machines running in different environments to communicate HTTP protocol.

2.3 Techniques used for Stock Price Prediction

In this study, two techniques of soft computing namely rough set theory and artificial neural network have been applied.

2.3.1 Methodology of Rough-Set Theory

Rough Set Theory, proposed in 1982 by Zdzislaw Pawlak. The RST does the approximation of lower and upper spaces of a set, the approximation of spaces being the formal classification of knowledge regarding the interest domain [7]. RST can be applied for the reduction of data to a minimal representation. In this technique dataset is represented as a table known as information table. In this table each row indicates an object (In this case, stock data for an instance of time) [8]. The RST technique works by finding the relevant attributes and eliminating the irrelevant attribute which are not essential. Every column represents an attribute [10].

2.3.2 Methodology of ANN technique

Artificial neural networks refer to a computing technique whose central theme is borrowed from the analogy of biological neural networks [9], [10]. In this soft computing technique previous known data are used to train the system and then

prediction is done using the trained system. In this study, Feed Forward Neural Network (FFNN) is used to predict the stock price.

A FFNN consist of neurons that are ordered in layers. The first layer is called the input layer, the last layer called the output layer, and the layers between these two layers are known as hidden layers. Each neuron in a particular layer is connected with all neurons in the next layer [11]. The interconnections between artificial neurons are called weights. The back propagation learning algorithm is applied for the training of the FFNN. Figure 2.1 shows a graphical representation of an artificial neural network. In Back-propagation algorithm computed error is used to update the weights between the layers to minimize the error which indicate the difference between the actual and computed value.

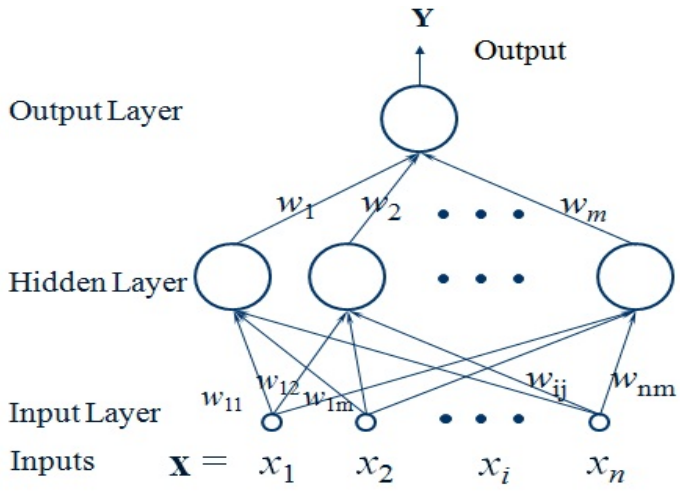

Figure 2.1: A Typical Single Layer FFNN

2.3.2.1 Training Phase:

In this phase the neural network is trained for improving quality. The procedure are as follows.

- **Input Layer Computation :** Input data are fed into the input layer. The output of the input layer is the input of the input layer.
- **Hidden Layer Computation:** The weights between the input and hidden layer are multiplied with the output data from the input layer and then the

results of multiplication from all the neurons connected to the hidden layer neuron, j are summed, which is fed into activation function to get the input into the hidden layer neuron, let, j; as shown in Equation 2.1 and Equation 2.2. In this study, sigmoid function is considered as the activation function, represented in equation 2.2.

$$a = \sum x_i w_{ij} \tag{2.1}$$

Where, x_i represents the inputs to the neuron and w_{ij} represents the weights of the neuron.

$$f(a) = \frac{1}{1 + e^{-a}} \tag{2.2}$$

- **Output Layer Computation :** Hidden layer neurons work in the same way to produce the output in the output layer. The sigmoid function is used as activation function. Output from activation function in the output layer is the expected output for the given input data.

- **Computation of Error :** This computed output from the neural net is compared with the actual output to compute the relative error. Calculation of the Error is shown in equation 2.3.

$$error_i = |(actualOutput_i - computedOutput_i)| \tag{2.3}$$

where,
$error_i$: error for the i^{th} data point
$actualOutput_i$: actual output for the i^{th} data point
$ccomputedOutput_i$: output computed from the neural net for the i^{th} data point.

- **Updating of Weights :** For every data point, error is calculated and then the weights are updated based on this relative error. Weights are updated according to the equation 2.4 and 2.5 [12]

$$W_{new} = W_{old} + [\Delta W]_{new} \tag{2.4}$$

$$[\Delta W]_{new} = -\eta \frac{\partial E}{\partial W} + \alpha [\Delta W]_{old} \tag{2.5}$$

where,

η : learning rate

α : momentum coefficient. $[0 \leq \alpha \leq 1]$

E : error

This whole process is repeated for every data point. When all data points in the training set are considered, one epoch is completed. For a training set having N number of data points, in every epoch Mean Square Error(MSE) is computed. MSE is computed using Equation 2.6. Epochs are repeated until MSE reaches a threshold value or the number of epochs reach the limit of maximum number of epochs.

$$MSE = \frac{\sum_{i=1}^{N} error_i^2}{N} \tag{2.6}$$

where,

N : Total number of data points.

2.3.3 Testing Phase :

Testing is done using the known dataset that is the corresponding output for the given input. For testing phase weights will not be updated. After the training phase is finished the last updated weights of the net are stored as the final weights. For testing, the inputs are fed and corresponding outputs are stored.

2.4 Stock Price Prediction Implementation

Stock price data are multidimensional in nature and possess the data on such as those opening stock price, closing stock price, volume of the stock, high stock price and low stock price. SOA implementation is carried out by using JBI framework in OpenESB tool. OpenESB helps to build, integrate and run the SOA applications

[13]. Stock data are collected using SOA application which is created, deployed and tested using OpenESB tool. This application uses `http://www.webservicex.net` web link. This application uses a WSDL file named as *stockquote.asmx.wsdl* for gathering the stock data. In this study, current stock data of two companies i.e. M/S Apple USA and M/S SBI India, have been collected as case study.

Figure 2.2 depicts the Business Process Executive Language (BPEL) process for orchestrating the services involving in stock price collecting application and a composite-application, for deploying and testing of a application are created. To test the composite-application, a test case is created that contains two XML based SOAP message, request and response respectively. The request-SOAP message, takes the symbol as input and corresponding response i.e., SOAP message returns the response after processing the request which are shown in Figure 2.3. This test program is executed to record the variation of stock price data for any day taken into consideration.

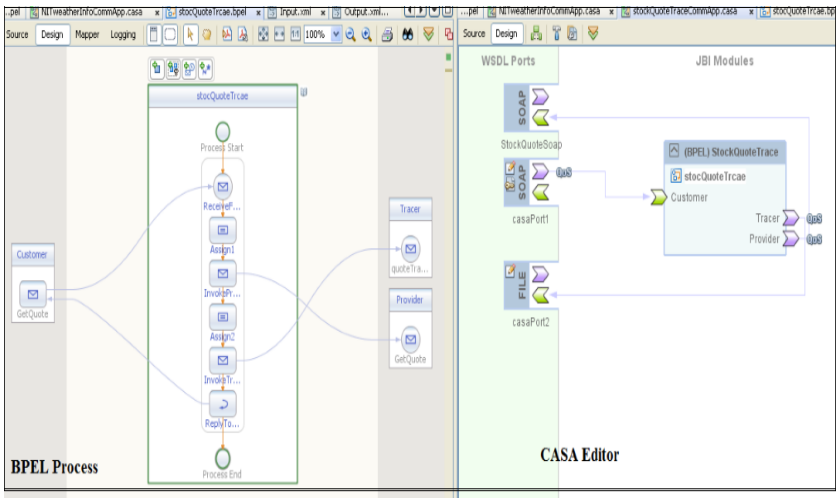

Figure 2.2: BPEL and CASA for Stock Data Collection Application

The sample of final attributed data which are being used for stock price prediction are shown in Table 2.1. The data are categorized as the closing, volume, opening, high value and low value. The high stock price of a company for next day is predicted using soft computing techniques.

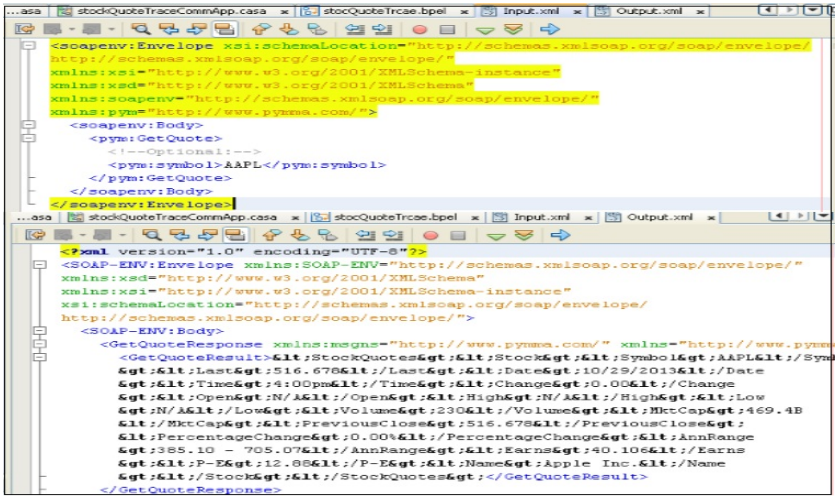

Figure 2.3: Request and Response SOAP Request Message for Apple stock quote

Table 2.1: Stock Sample of M/S Apple, USA

Date	Closing	Volume	Opening	High	Low
10/10/2011	388.81	15734430	379.09	388.81	378.21
11/10/2011	400.29	21598280	392.57	403.18	391.5
12/10/2011	402.19	22191070	407.34	409.25	400.14
...	...	...	...	...	...
07/10/2013	487.75	11127030	486.56	492.65	485.35
08/10/2013	480.94	10322930	489.94	490.64	480.54
09/10/2013	486.588	10322930	484.64	487.79	478.28
10/10/2013	489.638	9883786	491.32	492.38	487.04

2.4.1 Collection and Transformation of Stock Price Data

Table 2.1 indicates that collected stock data has total 5 attributes. Data are discretized in order to get the discrete dataset. The observation for any attribute in stock data has a large number of possible outcomes, hence it is quantized into smaller number of discrete values. First, the maximum data value **b** and minimum data value **a** for the attribute are computed. Then the overall interval is subdivided into a discrete set of subintervals. Number of intervals is computed using ***Sturges*** formula given in equation 2.7 [14].

$$k = log_2(n + 1) \tag{2.7}$$

where n is number of observations. The width of subintervals are calculated as equation 2.8 .

$$W = \frac{b-a}{k} \tag{2.8}$$

The interval boundaries are at *a+W, a+2W, ..., a+(k-1)W.*

2.4.2 Dimension Reduction Technique

Dimension reduction technique is required to reduce the number of attributes. ROSE tool has been used to apply RST on dataset [15]. First data is discretized and then it is supplied as input to ROSE tool. Initially the stock data has five attributes. After applying the RST algorithm number of attributes is reduced to four for M/S Apple and M/S SBI. In this study, closing stock price is predicted: so it is used as the target attribute as it should not be eliminated at the time of reduction.

2.4.3 Normalization

In ANN technique input and output values range between (-1, 1). So the data needs to be normalized within this range. Normalization process maps range of original data into another scale. In this case data are normalized in the range of (0, 1). The minimax normalization algorithm is used for normalizing the dataset [16]. The algorithm works as follows:
For a data vector $X(x_1, x_2, x_3, ..., x_n)$

- the maximum value, *Max* is found out.
- the minimum value, *Min* is found out.
- the maximum and minimum value of the normalized data are termed as *NewMax* and *NewMin* respectively which indicate the new range of the data.
- For any value x_i the normalized x_i indicated as $\bar{x_i}$ is computed as

$$\bar{x_i} = NewMin + \frac{(Xi - Min)(NewMax - NewMin)}{(Max - Min)} \tag{2.9}$$

2.4.4 Prediction using ANN Technique

ANN technique is applied to the normalized data for prediction of future temperature. Different steps for predicting the stock price are described below.

2.4.4.1 Training and Testing

FFNN technique is considered with a single hidden layer. Number of neurons in input layer, hidden layer and output layer is shown in Table 2.3. The normalized data are fed to the neural network for training.

Table 2.2: Testing and Training Details

Observation	Training Data		Testing Data	
	From	To	From	To
APPLE	October 10,2011	May 17,2012	May 20,2012	October 10,2013
SBI	October 18, 2011	May 17, 2013	May 20, 2013	October 9,2013

After the training of the neural network it is tested to find the accuracy of the training algorithm. Details of the training and testing periods are given in Table 2.2. Details of the parameters for the training are given in Table 2.3.

2.4.5 Prediction

When the neural net is trained, it is expected to give output with least error. For prediction if inputs are given for which output is not known; the net is expected

Table 2.3: Parameters considered for Training

Parameters	New Delhi	Calcutta
Neurons in Input Layer	4	4
Neurons in Hidden Layer	6	6
Neurons in Output Layer	1	1
Learning Rate	0.1	0.1
Momentum Coefficient	0.3	0.3
Maximum Number of Epochs	2000	2000

to predict the most accurate output. The closing stock price of the next day is predicted where previous day's stock price information is given as the input.

2.5 Performance Criteria

The following criterion are considered to evaluate the performance of algorithm used for prediction [17].

A. **Root Mean Square Error (RMSE):** Root mean square error is used as the performance criterion and calculated as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} error_i^2}{N}} \tag{2.10}$$

Where, N: Number of data points

B. **Mean Absolute Error (MAE):** The mean absolute error (MAE) is calculated as follows [18].

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|actual_i - predicted_i| \tag{2.11}$$

C. **Mean Percentage Error (MPE):** Mean percentage error (MPE) is the mean of percentage errors.

$$MPE = \frac{100\%}{N}\sum_{i=1}^{N}\frac{|actual_i - predicted_i|}{actual_i} \tag{2.12}$$

D. **Co-relation Co-Efficient (r):**

The Co-relation co-efficient, r measures relationship between two variables [19].

$$r = \frac{N\sum xy - \sum x \sum y}{\sqrt{[N\sum x^2 - (\sum x)^2][N\sum y^2 - (\sum y)^2]}} \tag{2.13}$$

where, x is the predicted value and y is the actual value.

E. **Mean Magnitude of Relative Error (MMRE):** MMRE is also used to evaluate the performance of the prediction technique. It measures the difference between the actual and predicted value to the actual value.

$$MMRE = \frac{1}{N}\sum_{i=1}^{N}\frac{|actual_i - predicted\ temp_i|}{actual_i} \tag{2.14}$$

F. **Accuracy:** Accuracy is the degree of matching between the predictions and the actual data. It is calculated as

$$Accuracy = (1 - MMRE) * 100\% \tag{2.15}$$

2.6 Results and Accuracy

In this study, closing stock price of a company is predicted of M/S Apple and M/S SBI which are shown in Figure 2.4 and Figure 2.5. The accuracy and errors for the prediction are shown in Table 2.4.

Table 2.4: Result and Accuracy

Accuracy Parameters	Apple	SBI
Threshold MSE	0.001	0.001
RMSE	0.0241	0.1195
MAE	0.0177	0.0867
MMRE	0.0247	0.1060
Accuracy	97.5327	89.3981
r	0.9731	0.9758

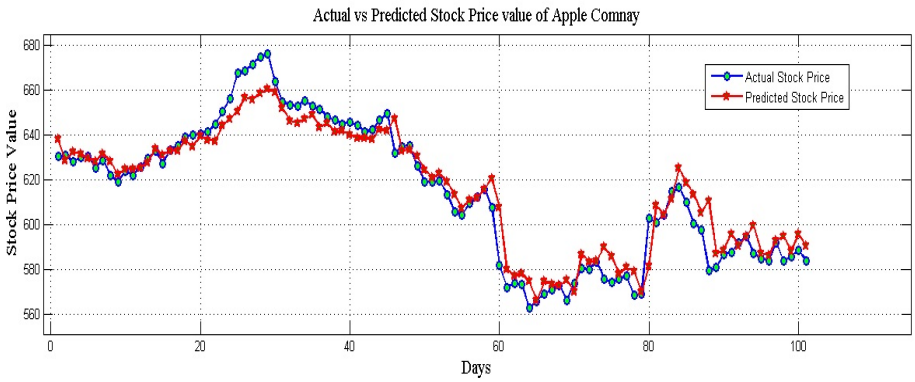

Figure 2.4: Predicted vs Actual Stock Price M/S Apple

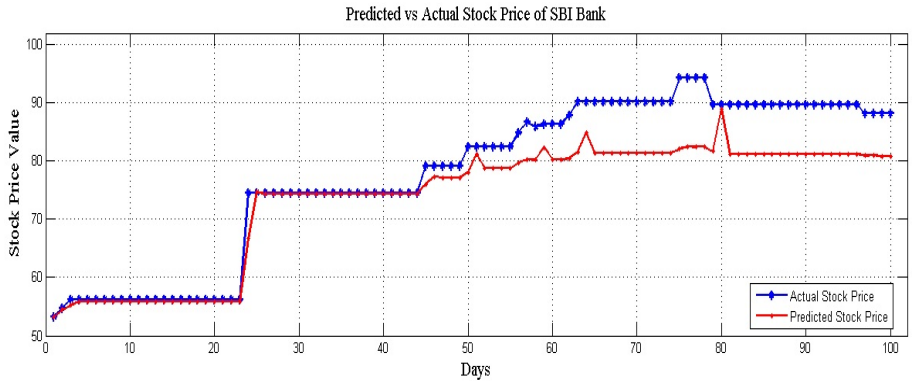

Figure 2.5: Predicted vs Actual Stock Price M/S SBI

2.7 Summary

In this Chapter 2, Service Oriented Architecture concept has been applied in order to explore the aspects of re-usability, interoperability and integration of data collection process. This study shows how the already created service, which is located at any remote location can be accessed, reused and integrated with application being carried out at any local system. SOA has been applied for collecting the stock price data. Attributes of data are curtailed using the technique of Rough Set Theory and then Artificial Neural Network model is further applied for prediction purpose. It is observed that the news articles play a very crucial role on any companys stock price. Future work of the study intends to include the impact of uncertainty in news article while predicting of stock price, by using the concept of service oriented architecture.

Chapter 3

Design and Implementation of Bank ATM based on Service Oriented Architecture and its Validation with Petri net model

This chapter describes how the Bank Automated Teller Machine(ATM) is designed and implemented using SOA concept. Further the verification of the process is done using Petri-net model.

An Automated Teller Machine (ATM) is an embedded system that is highly distributed in nature. It is a real-time application and includes a central bank server and a centralized account database. Application software of such system are desired to be responsive to unplanned changes in operational or business conditions. Existing literature states that the Service-Oriented Architecture(SOA) approach is one of the most effective strategies employed for solving the heterogeneous application integration problem. System developed using the concept of can reduce cost of overall development, time and risk as existing service helps in development. SOA supports in developing the new functions and services, integrated with the existing system, resulting in an expanded application. Selected functions can be progressively moved into a new environment, if there is a business case. This study introduces the application of concepts of SOA in automation and optimization of business processes, especially in the manner of communication enabled business processes. The ideas of SOA is widely accepted as fundamental in enhancing productivity and reliability associated with information technology. In

this study, various functionalities associated with ATM are implemented by using the concept of SOA and subsequent orchestration of the services is implemented by using Business Process Executive Language(BPEL) module under OpenESB framework. The correctness of this orchestration process i.e. BPEL process has been verified by using Petri net model. The verification ensures the correctness of linking of various services and message passing between them.

3.1 Introduction

ATM is an embedded systems for financial-oriented services that provide the customers of a bank or any financial institutions an access to financial transactions i.e. check balance, cash withdraw and transfer etc. in a public space without the need for a human clerk or bank teller. The architecture of a system of ATM is composed of individual ATM that are networked to a *HostComputer* and *BankComputer* as shown in the Figure 3.1. The host server is connected to other servers by Internet which makes it possible to all other ATM networks become available to the card holder.

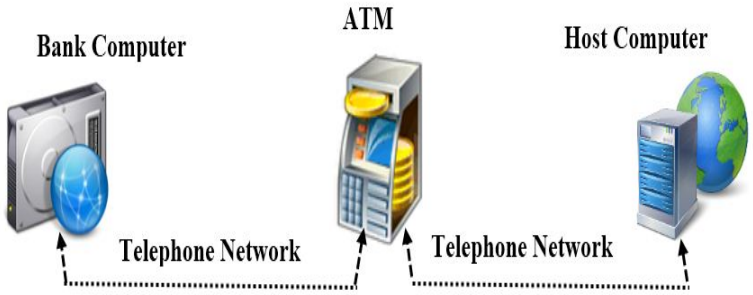

Figure 3.1: ATM Architecture

Present bank ATM terminology means that a person or business is authorizing another person or business to draft on an account. In this overall process, efficiency of ATM management, reliability, and security play an important role in producing a positive

SOA in bank ATM system provides a lot of benefits, such as improvement through re-usability and maintainability, abstraction, interoperability, and easy

use of resource-hungry services provided by more powerful Internet servers.

In this research, ATM system is implemented using the concept of SOA in which services interact to different services which are deployed at various component involved in the ATM transaction. By this the implementation process of ATM transaction can be more reliable, fast, and secure. More services can be added to ATM system if required as per the market demand. So with help of SOA concept agility of financial business can be handled easily and more efficiently. To ensure the correctness of various services orchestration can be guaranteed by formally verification of BPEL process of service by using petri net model.

3.1.1 Proposed SOA based Architecture of ATM system

As per SOA based ATM Architecture, ATMs, central database server, application server, mobile client server, commerce server and various banks are interlinked to each other by Enterprise Service Bus. The interaction among these components are based on messages which pass through Enterprise Service Bus(ESB). ESB is an integration platform that connects the various service layers and provides consumers with access to services. ESB provides the various functionalities i.e. validation of request and response, service binding, transformation of protocol and data, composition of elementary service to compote service, load balancing, security and service routing. Services are hidden behind the ESB and not directly visible to consumers. When a request from a service consumer is received, the ESB routes it to the correct service provider. This way, changes to services can be implemented without service consumers being aware. Sometimes, more than one service is available to fulfill the service request. The ESB can apply content based routing to determine which service the request needs to be routed to. The ESB flow first validates the inbound request message sent by consumers, transforms the message into a format that is understood by the service provider (e.g. transformation from one XML structure to another), and then routes the message to the appropriate service provider [20].

The work-flow adopted in this study, is shown in Figure 3.2

This study has proposed an SOA based ATM implementation in which services

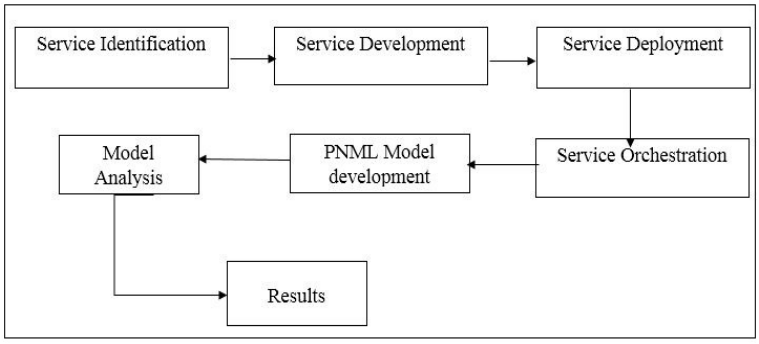

Figure 3.2: Work-flow Adopted in This Study

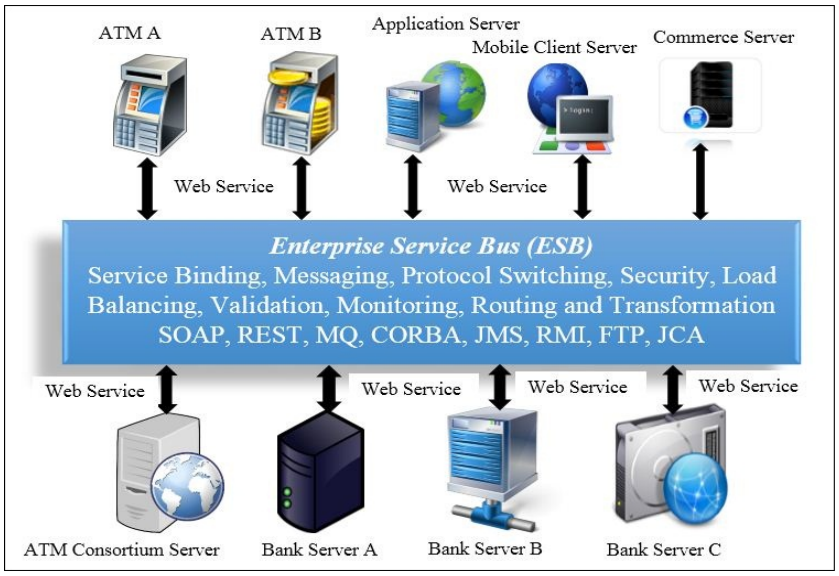

Figure 3.3: Proposed SOA based ATM Architecture

invokes demanded service automatically to fulfill the desired goal. The orchestration of these services further has been verified formally using petri net model to ensure the correctness of this process.

3.2 Literature Survey

Kart *et.al.* [21] discussed about application of SOA for distributed e-Health-care system. They claimed that large-scale distributed systems, i.e. as e-health-care systems, are difficult to develop as they are complex very complex in and decentralized in nature. Anthers presented a e-health-care system that uses the SOA as a basis for designing and implementing the e-health-care services.

In paper [22] Vasilescu *et. al.* discussed about implementing health-care system

by using the concept of SOA. In this paper authors explores the Integrating the Health-care Enterprise(IHE) efforts to improve the integration and interoperability of large scale health-care system.

S.Hinz [23] discussed Petri net model for the Business Process Execution Language for Web Services (BPEL). The authors discussed about semantics that covers the standard behavior of BPEL as well as the exceptional behavior (e.g. faults, events, compensation). They have used a tool known as LoLA for BPEL to petri net transformation.

In paper [24] Ouyang *et. al.* explain how to transform BPEL process to Petri net model and further its analysis. The authors discussed that in BPEL, the flow of logic of the service interactions between a different services is composition of communication actions which are correlated with control-flow dependencies expressed through constructs found in work-flow definition languages.

In this study, design and implementation of ATM system has been considered by using the concept of SOA and the orchestration of service is done by using BPEL. Further to formally verification of BPEL process is done by using Petri net model to ensure the correctness of BPEL.

3.3 Fundamentals of Techniques Used

In this study, the concepts of Service-Oriented Architecture is used for developing the ATM application. First of all, the services are identified and developed using OpenESB tool. To make a fully-furnished service like *Withdraw*, a number of web services i.e. *CardValidation*, *PINValidation*, *CheckBalance* etc. are orchestrated using BPEL module of OpenESB. Further, to check the correctness of BPEL process, Petri nets are used to verify.

3.3.1 Service Oriented Architecture

SOA can be used to solve the problem of how to connect disparate systems in a way that they could function together in a systematic manner [20]..

SOA is used to design and implement bank ATM.

3.3.2 Petri Nets

Petri nets are one of the several formal models used extensively for verification and validation. A Petri net is a bipartite directed graph which contains three types of entities such as places, transitions, and directed arcs [25]. Petri nets have strong mathematical nuts and bolts with a graphical representation. Automatic analysis and verification techniques are applied by the use of mathematical foundation. Petri nets can be best applied in order to design complex large system to check the concurrency, synchronization, deadlocks, and resource utilization among systems. There are a number of tools based on Petri net theory for analysis of models that help the users to graphically analyze a model, simulate them through an animated sequence and use them to validate the process. The performance of various models are compared by using Platform Independent Petri net Editor (PIPE) tool [26].

PIPE tool is a Stochastic Petri net tool in which randomizing delays are associated with transitions. The generalized stochastic Petri net (GSPN) tool is also used to evaluate the performance measures [27]. In the performance analysis, average number of tokens and throughput of a transition are measured.

An Stochastic Petri net (SPN) shown in first part of Figure 3.4 illustrate how to analyze the performance parameter [28]. Initially Transition t_1 is enabled at marking $M_0 = (1, 0, 0, 0, 0)$ and time elapsed until t_1 fires is exponentially distributed with rate λ_1, so the average time for t_1 to fire is $1/\lambda_1$. Once t_1 is fired, marking $M_1 = (0, 1, 1, 0, 0)$ is available according to the firing rule of Place-Transition nets. At M_1, t_2 and t_3 becomes concurrently enabled which means one of these two transitions will fire next after a certain duration. In case, if transition t_2 will fires first, then SPN will changes to marking M_2 = (0,0,1,1,0). And if t_3 fires earlier to t_2, it change to the marking M_3 = (0,1,0,0,1). Thus next marking will depends on which transition fires first. The probability that t_2 fires first is given by equation 3.1: [28].

$$P\left[t_2 fires first at M_1\right] = P\left[X_2 < X_3\right]$$
$$= \int_0^\infty \left(\int_0^x \lambda_2 e^{-\lambda_2 y} dy \right) \lambda_3 e^{-\lambda_3 x} dx$$

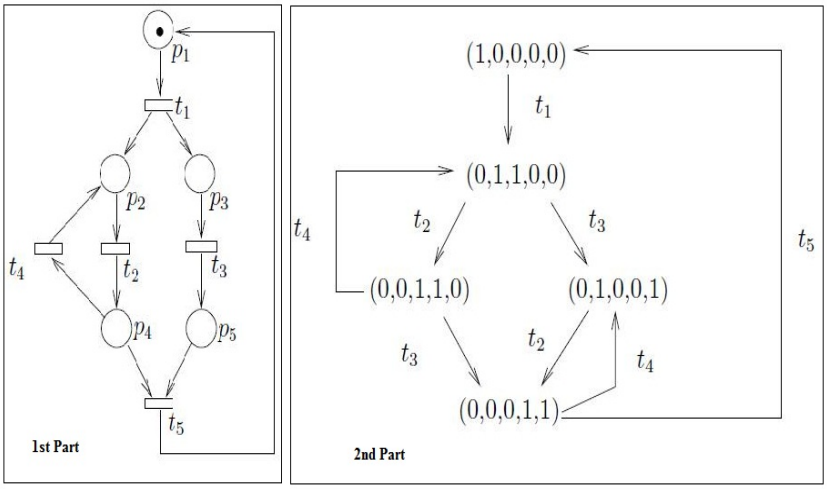

Figure 3.4: Stochastic Petri net and its corresponding Reachability Graph

$$= \int_0^\infty \left(1 - e^{-\lambda_2 x}\right) \lambda e^{-\lambda_3 x} dx$$

$$= \lambda_2/(\lambda_2 + \lambda_3) \tag{3.1}$$

and similarly, the probability of firings of T_3 can be represented as per equation 3.2

$$P\left[t_3 fires first at M_1\right] = \lambda_3/(\lambda_2 + \lambda_3) \tag{3.2}$$

It can be observed here that the probability of changing from marking M_1 is independent of the time spent in M_1.
Sojourn time in marking M_1 is calculated by equation 3.3 that the minimum of the independent and exponentially distributed firing times of both transitions,

$$\begin{aligned} P\left[min\left(X_2, X_3\right) \leq x\right] &= P\left[X_2 \leq x \; or \; X_3 \leq x\right] \\ &= 1 - P\left[X_2 > x \, and \, X_3 > x\right] \\ &= 1 - e^{-\lambda_2 x} e^{-\lambda_3 x} \\ &= 1 - e^{-(\lambda_2 + \lambda_3)} \end{aligned} \tag{3.3}$$

This can be observed here, that sojourn time in M_1 is also exponentially distributed with parameter $(\lambda_2 + \lambda_3)$. In Figure 3.4 second part depicts the reachability graph of first part of Figure 3.4. Markov chain can be obtained by attaching

the firing rate λ_i of transition i as an arc label to each transition as shown in second part of Figure 3.4.

The Markov chain for an SPN can be achieved with the help of the reachability graph. State-space is the reachability set R(PN) and the transition rate from state m_i to state m_j is given by $q_{ij} = \lambda_k$. In case of several transitions lead from m_i to m_j then q_{ij} is the sum of the rates of those transitions.

The steady state distribution π of the Markov chain is calculated by solving the equations 3.4.

$$\pi Q = 0 \tag{3.4}$$

$$\sum_{i=1}^{s} \Pi_j = 1 \tag{3.5}$$

where,

Q is the square matrix

Probability of being in a subset of markings:

The probability of being in a state of the corresponding subset of the Markov Chain is given by equation 3.6.

$$P\left[B\right] = \sum_{M_i \epsilon B} \pi_i \tag{3.6}$$

where,

$B \subseteq R(PN)$ is marking of interest in a particular SPN.

Mean number of tokens:

Then the average number of tokens in place p_i is given by:

$$\bar{m_i} = \sum_{n=1}^{\infty} \left(nP\left[B\left(P_i, n\right)\right]\right) \tag{3.7}$$

where,

$B(p_i, n)$ be the subset of $R(PN)$

Probability of firing transition t_j:

Let EN_j be the subset of R(PN) in which a given transition t_j is enabled. Then, the probability r_j for transition t_j firing next is given by:

$$r_j = \sum_{M_i \epsilon EN_j} \pi_i \left(\lambda_j \setminus (-q_{ij})\right) \tag{3.8}$$

where $(-q_{ii})$ is the sum of transition rates out of M_i.

Throughput at a transition t_j:

At a timed transition, throughput is given by it's mean number of firings at steady state:

$$\bar{d_j} = \sum_{M_i \epsilon EN_j} \pi_i \lambda_j \tag{3.9}$$

The performance of the model can be investigated on the basis of the values of these parameters.

3.4 Description of Proposed Work

In the proposed SOA based ATM architecture, the various services are identified. In SOA based system, everything is based on services. Service is a logical encapsulation of self-contained business functionality and their identification is even important because deciding the service granularity decides the complexity and abstraction of application. In this study various services i.e. *LocalATM* service, *Card Validation* service, *PINValidation* service, *CheckBalance* service, *ChangePIN* service, *Withdraw* service, *TransferService*, *StatementGeneration* service, and *AccountUpdate* etc. service are identified, modeled, developed and deployed.

The identified services are deployed at the corresponding components as shhown in Figure 3.5.

LocalATMService is a service which starts the process of ATM transaction and is deployed at ATM Machine. *CardValidation* service is first service which is responsible for validating their ATM card and is deployed on the ATM Consortium Server. This service verifies the card based on card information scanned by ATM Machine and it takes the help of customer verification service of customers bank. *PINValidation* Service is the service which responsible for validation of the PIN number entered by the customer and is deployed at customers bank server.

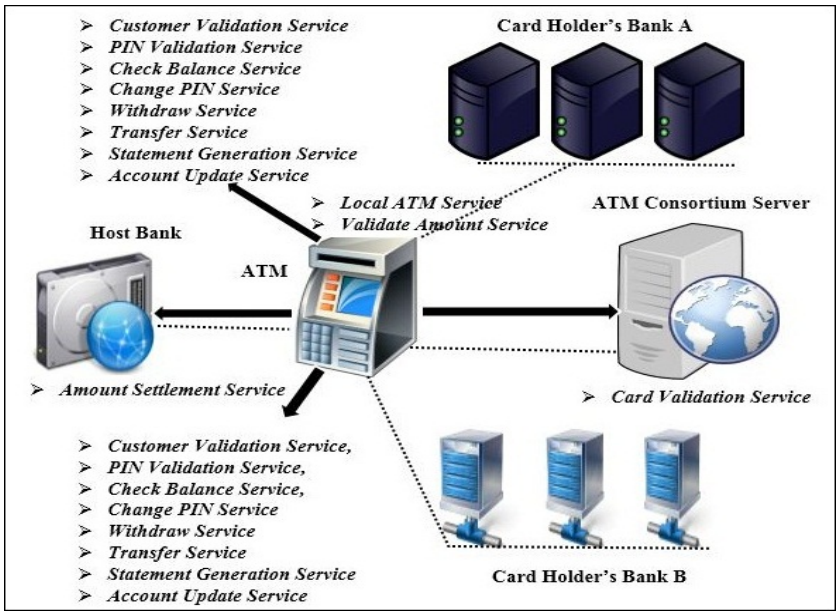

Figure 3.5: ATM Components with their Services

ChangePIN Service, *Withdraw* Service, *Transfer* Service, *StatementGeneration* Service, and *AccountUpdate* Service are also deployed at customers bank server and perform their corresponding responsibilities. The study assumes that the corresponding services are deployed at all banks associated with ATM consortium and service of that bank is invoked to which the customer belongs to. This makes the automatic interaction of applications of different banks. Also, these services can be reused in new applications like Mobile Banking, Internet Banking etc.

For developing a particular service(like *Withdraw*, *CheckBalance*), a number of web services are orchestrated. This is done with the help of Business Process Execution Language(BPEL). The web services are created using an open source tool e.g. OpenESB. The SOA-based GUI can be created using a WSDL file which is used as an input to BPEL process. But the BPEL model(.bpel file) cannot be directly used for formal verification. Thus, convert the .bpel file to a PNML model(.pnml file) first. This is done with the help of BPEL2PNML tool. PIPE2 is used to formal verification of BPEL process (.bpel file).

3.5 Experiment Details

3.5.1 Case study-Withdraw Scenario based on SOA

When a customer inserts the ATM card into the ATM machine, the card reader reads the information of the card i.e. card number. The ATM Local service invokes the *CardValidation* service of ATM Consortium which returns the bank of card issuer and account type if the card is valid or an error message if the card is invalid. Now the user is prompted to enter the PIN number and the amount to be withdrawn. First the service *AmountValidation*(deployed at ATM) is invoked because the entered amount has to be valid(multiple of hundred, less than the one-time withdraw limit, which is ten thousand in this case and less than the the current cash present in the ATM Cash pot) for proceeding with the withdrawal process; otherwise the user needs to re-enter a valid amount. Next, the *PINValidation* service is invoked to match the PIN number(it is a service deployed at the bank to which this customer belongs). If the PIN number is correct, then a service *MaxDayLimit*(deployed at card issuer bank) is invoked to check whether the card holder has exceeded the withdrawal limit for the current day. If yes, then the withdrawal process is not allowed and the customer is prompted to re-enter a smaller amount. After this, another service *CheckMinBalance* is invoked to decide if there is sufficient balance available in that account. If no, then withdraw cannot take place so user is again prompted to re-enter a smaller amount. If these conditions are satisfied, then the ATM Local service invoke the customer bank's withdraw service with inputs: card number, host bank account number and amount to be withdrawn. This service first debits the withdrawal amount from the customer's account and electronically transfers it to the host banks account and records this transfer in the transaction database. This service returns the transaction ID to the host ATM, which is used by ATM local service to verify the successful amount transfer by invoking the service of host bank server i.e. *CheckFundTransfer* Service. If the electronic fund transfer was successful, then ATM dispenses the cash and displays the message to collect the cash. ATM invokes the *CheckBalance* service for displaying the remaining balance in the customer's

account. Finally, the cash of ATM pot is updated and this transaction is recorded in the database by invoking the *UpdateATMCash* Service. If due to some reason the customer could not get the cash, then the amount from the customer account needs to be credited back. This can be easily done in the SOA paradigm using a service(*CashClearance* Service). This service is depolyed at host bank server. In this scenario, it is observed that services interact with each other to achieve a desired goal. So with the help of SOA concepts, business processes can be fully automated. The orchestration of the services helps in defining the workflow of a particular business need.

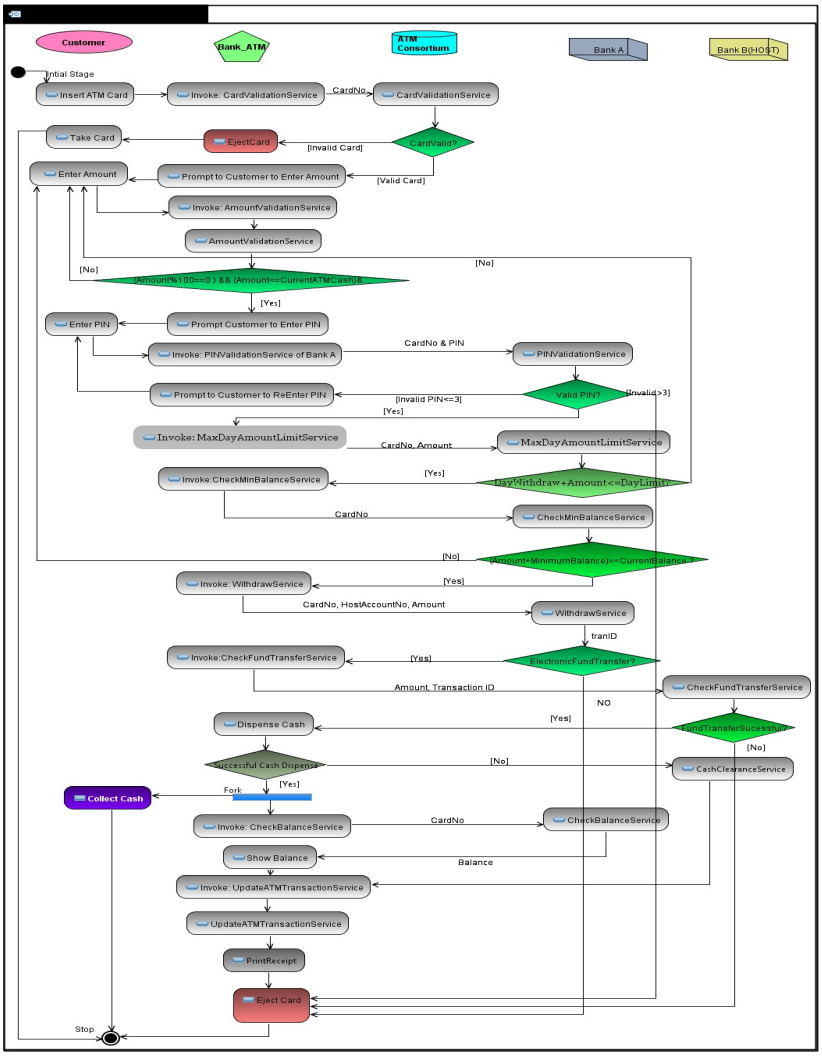

Figure 3.6: Activity Diagram of ATM Cash Withdraw Scenario

3.6 Development of BPEL Process for Withdraw Service of ATM

Once the services and operations are identified, one can develop these services by using any sort of tools and techniques available. Now a days services may be developed either based on SOAP or REST communication protocol. The selection of SOAP or REST protocol is decided based on the application in which these are going to be used. These services are self-contained and self-describing and can be discovered by using UDDI. HTTP and XML form the basis of these services. HTTP and XML make services interoperable and solution agnostic. These services are solutions as one service is design and developed independent of particular solution. Service can be reused in many applications. In this study all services have been developed based on SOAP by using the tool OpneESB. List of services identified, developed and deployed on different components as per SOA based ATM Architecture are shown in Figure 3.7.

Deployed Component Name			
ATM Machine Server	ATM Consortium Server	Banks Server	Host Bank Server
1. Amount Validation Service, 2. Update ATM Transaction Service, 3. Print Receipt Service	1. Card Validation Service	1. PIN Validation Service, 2. Max Day Amount Limit Service, 3. Check Min Balance, Service, 4. Withdraw Service, 5. Check Balance Service	1. Check Fund Transfer Service, 2. Cash Clearance Service.

Figure 3.7: List of Services Identified, Developed and Deployed on Different Components

3.6.1 Business Process Execution Language(BPEL)

It facilitates the orchestration of services. The properties of SOA like reusability and agility can be achieved with the help of BPEL. In this study, some services have been used for both check balance and withdrawal purpose.The BPEL of check balance uses the services like *CardValidation*, *PINValidation* and *CheckBalance*.

SOAP Request Message:

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Header/>
    <S:Body>
        <ns2:sbiwithdraw xmlns:ns2="http://sbi/">
            <CardNumber>5196190031114269</CardNumber>
            <HostAccountNo>14580001000046162</HostAccountNo>
            <Amount>10000</Amount>
        </ns2:sbiwithdraw>
    </S:Body>
</S:Envelope>
```

SOAP Response Message:

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:sbiwithdrawResponse xmlns:ns2="http://sbi/">
            <TranID>123457</TranID>
        </ns2:sbiwithdrawResponse>
    </S:Body>
</S:Envelope>
```

Figure 3.8: SOAP Request and Response Message for Withdraw Service

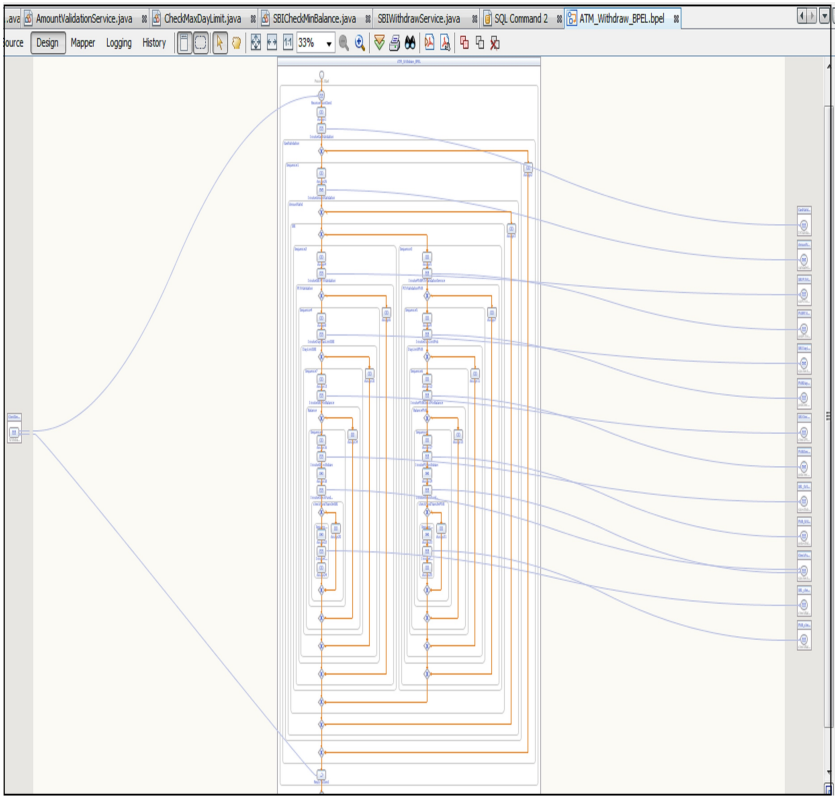

Figure 3.9: bpelWithdraw

The orchestration for cash withdraw scenario, which is shown in Fig 3.9, uses the services like *CardValidation*, *AmountValidation*, *PINValidation*, *MaxDayLimit*, *CheckMinBalance*, *Withdraw*, *ElectronicFundTransfer*, *CheckFundTrans-*

fer, *CashClearance*, *CheckBalance*, *UpdateATMTransaction*. Thus here three services have been reused i.e. *CardValidation*, *PINValidation* and *CheckBalance*. This implies that SOA can handle the problem of business agility. The services that were already developed can be reused directly or with slight modifications as per agile business requirement with the help of BPEL process.

3.7 Transforming BPEL Process to Petri net

We have the BPEL file of *Withdraw* Now first the .bpel file converted into PNML by using the BPEL2PNML.jar file,for this purpose command used is shown in Figure 3.10.

```
F:\>cd BPEL2PNML

F:\BPEL2PNML>java -jar BPEL2PNML.jar g ATM_Withdraw_BPEL.bpel
```

Figure 3.10: BPEL to PNML Conversion Command

Petri net model of Withdraw Service is shown in Figure 3.11

3.8 Results and Analysis

BPEL is XML based language for composition of service a widely used by business process developers . Thereby, it is important to be able to verify business processes and determine the inconsistencies of the language's specifications presented in describing the BPEL process. BPEL lacks mathematical semantics, and therefore BPEL specifications are formally verified by translating them into Petri nets to facilitate their analysis.

After transforming the BPEL process into Petri nets, the performance analysis of the Petri nets model is carried out using PIPE tool. The PIPE tool analyzes the steady state distribution of tangible states, average number of tokens in each place, token probability density, throughput of timed transition and sojourn time of tangible states. PIPE tool also examines the boundedness, safeness and presence of deadlock in the model.

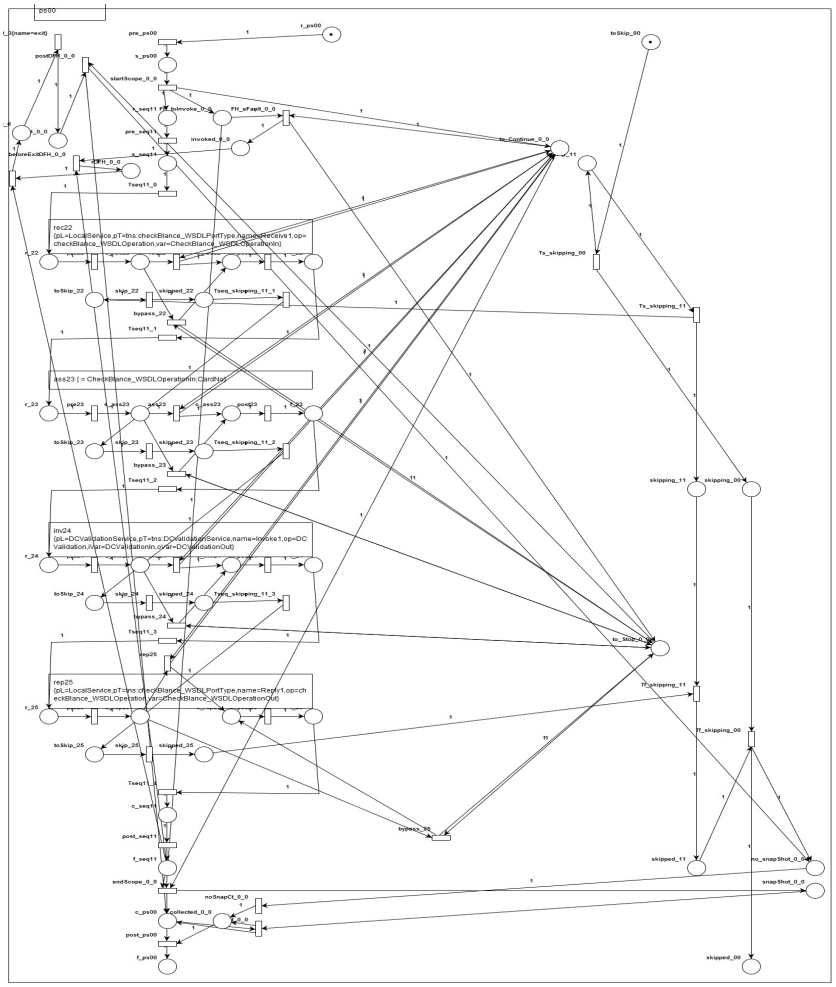

Figure 3.11: Petri net Model of Withdraw Service

3.8.1 Average Number of Tokens in a place

The following tables illustrates the performance analysis result of the BPEL process. Table 3.1 shows the average number of tokens present in a place. It can be calculated by the equation 3.7.

3.8.2 Token Probability Density

Table 3.2 specifies the token probability density and it can be calculated by the equation 3.9.

Table 3.1: Average Number of Token on a Place

Place	Number of Tokens
c_ass23	0
c_inv24	0
$c\_DFH\_0\_0$	0
$FH\_toInvoke\_0\_0$	1
$invoke\_0\_0$	0
$no\_snapShot\_0\_0$	1
.	.
.	.
.	.
$skipping\_00$	1
$snapShot\_11$	0
$to\_Continue_{0\_0}$	1
$skipped\_00$	0
$to\_Continue\_0\_0$	1
to_Stop_00	0
$toSkip\_11$	0
to_Stop_22	1
to_Stop_23	0
to_Stop_24	0
to_Stop_25	0

Table 3.2: Token Probability Density

Place	$\mu = 0$	$\mu = 1$	$\mu = 2$
c_ass23	1	0	0
c_inv24	1	0	0
c_ps00	1	0	0
f_ps00	0.00001	0.00001	0
$FH\_toInvoke\_0\_0$	0	1	0
$invoked\_(0\_0)$	1	0	0
.	.	.	.
.	.	.	.
.	.	.	.
$no\_snapShot\_0\_0$	0	1	0
f_seq11	1	0	0
f_se11	1	0	0
$to\_Continue\_0\_0$	0	1	0
$to\_Stop\_0\_0$	1	0	0
$toSkip\_25$	1	0	0

3.8.3 Throughput of Timed Transition

Table 3.3 shows the throughput of timed transition and calculated as equation 3.9.

Table 3.3: Throughput of Timed Transition

Transition	Throughput
$ass23$	0
$beforeExitDFH\_0\_0$	1
$bypass\_22$	1
$bypass\_23$	1
$bypass\_24$	1
.	.
.	.
.	.
$endScope\_0\_0$	1
$exitDFH\_0\_0name = exit$	1
$rec22$	0
$skip\_25$	1
$snapCt\_0\_0$	1
$post\_\_ps$	1
$post\_seq11$	1
T_seq11_4	1

3.8.4 Sojourn Time for Tangible States

Table 3.4 shows the sojourn time of the tangible states. Tangible states are those states which are associated with the timed transition.

Table 3.4: Sojourn Time for Tangible States

Marking	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	.	.	M_{15}	M_{16}	M_{17}	M_{18}	M_{19}
Value	0.03448	0.03448	0.03448	0.03448	0.03448	0.03448	0.03448	0.03448	0.03448	0.03571	0.5	.	.	0.03448	0.03571	0.03571	0.03571	0.03704

3.8.5 Reachability Graph of *WithdrawService's* BPEL

Visual representation of all the possible firing sequences for the given Petri net can be provided by Reachability Graph. It also gives information about safeness, boundedness deadlock-free properties.

Figure 3.12 shows the reachability graph of the model, which provides a visual representation of all the possible firing sequences for the given Petri net. It also gives information about boundedness (Buffer flow is not possible), safeness and deadlock-free properties. The firing of the enabled transition changes the state of the model. The sequence of firing results in getting a sequence of markings. This sequence of markings can be portrayed as a reachability graph. The reachability graph of BPEL process guarantees the correctness of connectivity between services involved in orchestration.

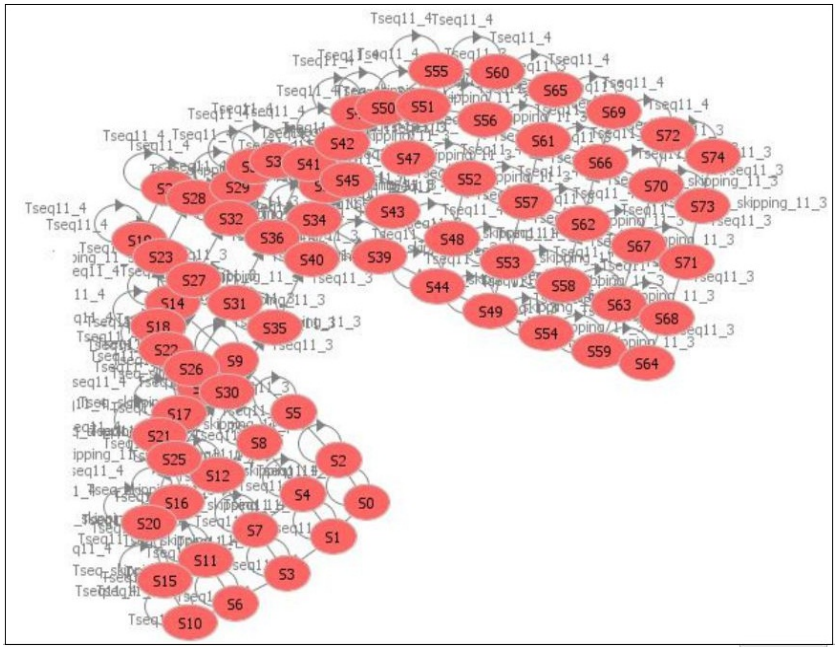

Figure 3.12: Reachability graph of Withdraw Service

3.8.6 State Space Analysis

To determine the qualitative properties of the given Petri net i.e. deadlock-free, safeness and boundedness a tree a built by State Space Analysis with all the reachable markings.

It also provides the shortest path to deadlock in the case that there exists one. From the state space analysis of the model obtained using PIPE tool, it is observed that the model is bounded, safe and deadlock free. The result of state-space analysis of BPEL process for ATM Withdraw is shown in Figure 3.13.

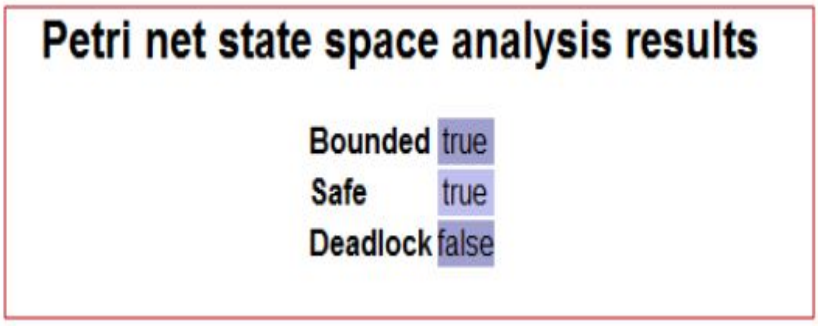

Figure 3.13: State Space Analysis

3.9 Summary

In this study, an ATM system is designed and implemented based on the concepts of SOA. Services are identified, designed and developed as per the proposed SOA-based architecture using OpenESB tool. Further, these services are orchestrated as per a particular business requirement i.e. Withdraw Cash from ATM in this case. In a similar manner, other business requirements like *CheckBalance*, *TransferCash*, *DepositCash* etc. can be implemented by re-using the existing services. The orchestration(BPEL process) is verified using Petri net model for ensuring it's correctness. The development cost and time can be reduced to a great extent by using SOA as it fosters reusability. Also, changes can be implemented quickly either by adding new services to the existing system or slightly modifying the existing services of the system, hence SOA based systems support enterprise agility.

Chapter 4

Conclusion and Future Work

Service Oriented Architecture has become the important design and implementation paradigm in todays' agile business scenario. SOA can use existing services placed at local or remote to any environment. Services can be designed, developed and deployed as per requirements. In first case, SOA concept has been applied in order to explore its aspects i.e. re-usability, interoperability and integration of data collection process with help of existing service located at `www.webservicex.net`. Here it is shown how the existing created service, which is located at any remote location can be accessed, reused and integrated with applications being carried out at any local system. SOA has been applied for collecting the stock price data. Attributes of data are curtailed using the technique of Rough Set Theory and then Artificial Neural Network model is further applied for prediction purpose.

In second case, an ATM system is designed and implemented based on the concepts of SOA. Services are identified, designed and developed as per the proposed SOA-based architecture using OpenESB tool. Further, these services are orchestrated as per a particular business requirement i.e. Withdraw Cash from ATM in this case. In a similar manner, other business requirements like *CheckBalance*, *TransferCash*, *DepositCash* etc. can be implemented by re-using the existing services. The orchestration(BPEL process) is verified using Petri net model for ensuring it's correctness. The development cost and time can be reduced to a great extent by using SOA as it fosters re-usability. Also, changes can be implemented quickly either by adding new services to the existing system or slightly modifying the existing services of the system, hence SOA based systems support enterprise

agility. Overall, the SOA concept is explored to take its advantage in enterprise business application's design and implementation. By applying this concept development cost and time can be reduced to a very high extent and enterprise agility can be handled with SOA efficiently.

Future work of the study intends to apply SOA for analysis of the Big data and in the development of the Internet of Things. Big data is the structured and unstructured information that a company accumulates in high volumes and at rapid velocity. Depending on how it's managed, big data can be a valuable resource or an expensive problem. RavenDB is a database like SQL, which handle the unstructured data. One service for handling unstructured data by using RavenDB and another service using SQL for structured database can be developed separately. Further with the help of SOA these services can integrated to handle Big data in an efficient way.

The Internet of Things (IoT) is a scenario in which things i.e. people, animals or objects are connected to the Internet with a unique identifiers and has the ability to automatically transfer data over a network without human intervention. In IoT, service-oriented architecture (SoA) might be imperative for the service providers and users. SoA ensures the interoperability among the heterogeneous devices in multiple ways. So the use of SOA can be extended in this future technology of IoT.

Bibliography

[1] T. Erl, *Service-oriented architecture: concepts, technology, and design*. Pearson Education India, 2005.

[2] D. Sprott and L. Wilkes, "Understanding service-oriented architecture," *The Architecture Journal*, vol. 1, no. 1, pp. 10–17, 2004.

[3] A. Brown, S. K. Johnston, G. Larsen, and J. Palistrant, "Soa development using the ibm rational software development platform: a practical guide," *Rational Software*, 2005.

[4] S. Mazumder, "Soa: A perspective on implementation risks," *SETLabs Briefings*, 2006.

[5] F. Wang, L. Liu, and C. Dou, "Stock market volatility prediction: A service-oriented multi-kernel learning approach," in *Services Computing (SCC), 2012 IEEE Ninth International Conference on*, pp. 49–56, IEEE, 2012.

[6] Y.-H. Wang, S.-C. Chen, and P.-H. Peng, "Applying service-oriented architecture to construct the banking letter of credit system integration," 2013.

[7] S. Rissino and G. Lambert-Torres, "Rough set theory–fundamental concepts, principals, data extraction, and applications," *Data Mining and Knowledge Discovery in Real Life Applications, J. Ponce and A. Karahoca (Eds.), InTech Publishers*, 2009.

[8] B. Ruzgar and N. S. Ruzgar, "Rough sets and logistic regression analysis for loan payment," *International Journal of Mathematical Models and Methods in Applied Sciences*, vol. 2, no. 1, pp. 65–73, 2008.

[9] S. Soni, "Applications of anns in stock market prediction: a survey," *International Journal of Computer Science & Engineering Technology*, vol. 2, no. 3, pp. 71–83, 2011.

[10] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural Networks, 1989. IJCNN., International Joint Conference on*, pp. 593–605, IEEE, 1989.

[11] I. Maqsood, M. R. Khan, and A. Abraham, "An ensemble of neural networks for weather forecasting," *Neural Computing & Applications*, vol. 13, no. 2, pp. 112–122, 2004.

[12] S. Rajasekaran and G. V. Pai, *Neural Networks, Fuzzy Logic and Genitic Algorithm: Synthesis and Applications (with CD*. PHI Learning Pvt. Ltd., 2003.

[13] open esb.net, "Open esb. the open enterprise service bus." `https://today.java.net/pub/a/today/2008/10/06/sending-sms-messages-with-jbi.html`.

[14] F. R. C. Nina, *On applications of rough sets theory to knowledge discovery*. PhD thesis, Citeseer, 2007.

[15] B. Predki and S. Wilk, "Rough set based data exploration using rose system," in *ISMIS*, vol. 99, pp. 172–180, 1999.

[16] J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*. Morgan kaufmann, 2006.

[17] T. Menzies, Z. Chen, J. Hihn, and K. Lum, "Selecting best practices for effort estimation," *Software Engineering, IEEE Transactions on*, vol. 32, no. 11, pp. 883–895, 2006.

[18] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance," *Climate Research*, vol. 30, no. 1, p. 79, 2005.

[19] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[20] L. Dikmans and R. Van Luttikhuizen, *SOA Made Simple*. Packt Publishing Ltd, 2012.

[21] F. Kart, G. Miao, L. E. Moser, and P. Melliar-Smith, "A distributed e-healthcare system based on the service oriented architecture," in *Services Computing, 2007. SCC 2007. IEEE International Conference on*, pp. 652–659, IEEE, 2007.

[22] E. Vasilescu and S. K. Mun, "Service oriented architecture (soa) implications for large scale distributed health care enterprises," in *Distributed Diagnosis and Home Healthcare, 2006. D2H2. 1st Transdisciplinary Conference on*, pp. 91–94, IEEE, 2006.

[23] S. Hinz, K. Schmidt, and C. Stahl, "Transforming bpel to petri nets," in *Business Process Management*, pp. 220–235, Springer, 2005.

[24] C. Ouyang, E. Verbeek, W. M. van der Aalst, S. Breutel, M. Dumas, and A. H. ter Hofstede, "Wofbpel: A tool for automated analysis of bpel processes," in *Service-Oriented Computing-ICSOC 2005*, pp. 484–489, Springer, 2005.

[25] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, Apr, 1989.

[26] S. Liu, R. Zeng, and X. He, "Pipe+ -a modeling tool for high level petri nets," in *proceedings of International Conference on Software Engineering and Knowledge Engineering (SEKE11)*, pp. 115–121, 2011.

[27] D. Kartson, G. Balbo, S. Donatelli, G. Franceschinis, and G. Conte, *Modelling with generalized stochastic Petri nets*. John Wiley & Sons, Inc., 1994.

[28] F. Bause and P. S. Kritzinger, "Stochastic petri nets: An introduction to the theory," *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, no. 2, pp. 2–3, 1998.

Dissemination of Work

Accepted

1. Amar Nath, Sumana Maity and Santanu Kumar Rath Stock Price Prediction Using Service Oriented Architecture and Soft Computing techniques. *in Proceedings of BAICONF: 1st IIMB International Conference on Business Analytics and Intelligence, held on 11-13 Dec 2013, IIM Bangalore, India.*

2. Amar Nath and Santanu Kumar Rath. Optimized Scenario of Temperature Forecasting using Service Oriented Architecture Coupled with Soft Computing Techniques. *in Proceedings of ERCICA: 2nd International Conference on Emerging Research in Computing, Information, Communication and Applications, ERCICA 2014, to be held on 01-02 Aug 2014 in Nitte Meenakhsi Institute of Technology, Bangalore, India.*

Communicated

3. Amar Nath, Prerna Kanojia and Santanu Kumar Rath. Service Oriented Architecture based integration of Telecommunication Services and Subsequent Verification using Petri-net. *in Proceedings of IC3: 7th International Conference on Contemporary Computing(IC3), to be held on 07-09 Aug 2014 Jointly Organized by Jaypee Institute of Information Technology and University of Florida At JIIT, Noida August 7-9, 2014.*

Chapter 5

Annexure

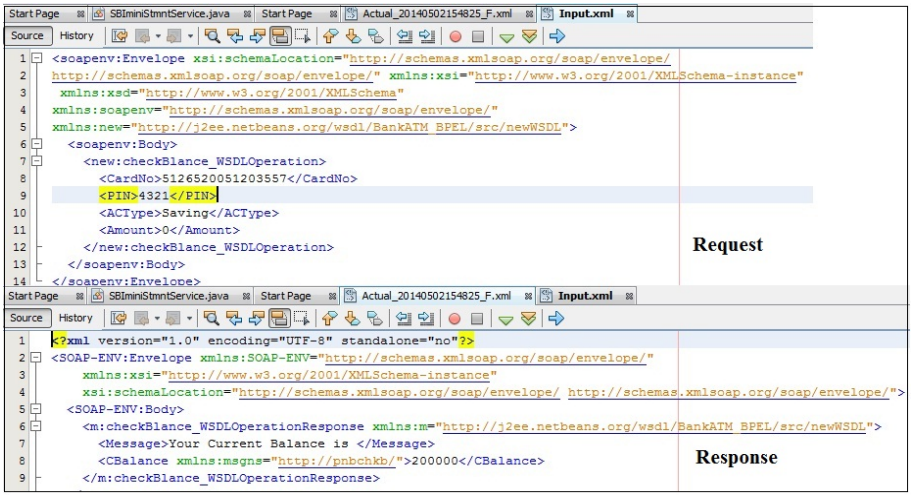

Figure 5.1: Request and Response of Final *CheckBalance* Service

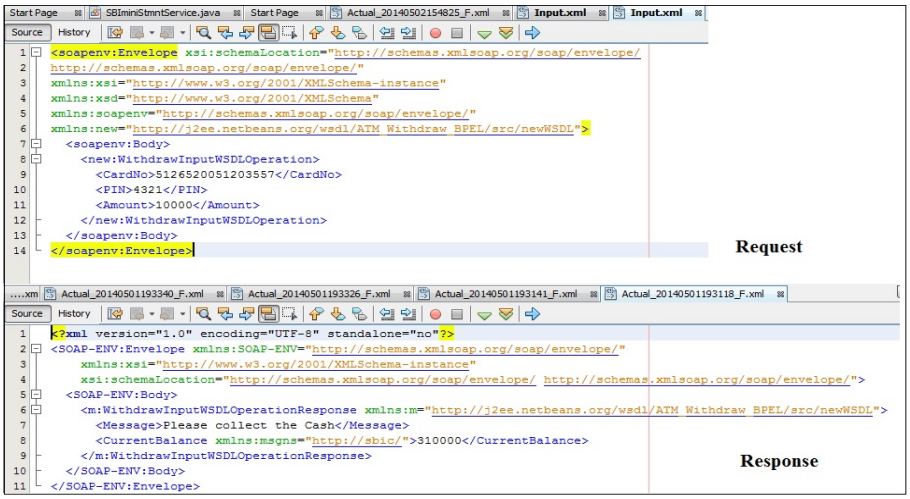

Figure 5.2: Request and Response of Final *Withdraw* Service

5.1 Source Code of Withdraw Service.

```
package sbi;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import javax.annotation.Resource;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
import javax.sql.DataSource;
import javax.xml.ws.Holder;

    @WebService(serviceName = "SBIWithdrawService")
public class SBIWithdrawService
@Resource(name = "atmDB")
private DataSource atmDB;
private String StID;
private int tranid;
private String Tid;
private String AccountNumber = "1111222233334444";
private double Amount1;

    @WebMethod(operationName = "sbiwithdraw")
public void sbiwithdraw(
```

@WebParam(name = ”CardNumber”) String CardNumber,

@WebParam(name = ”HostAccountNo”) String HostACNumber,

@WebParam(name = ”Amount”) Integer Amount,

@WebParam(name= ”TranID”, mode = WebParam.Mode.OUT)

Holder¡String¿TranID

)

try

Amount1 = (double)Amount;

Connection conn=atmDB.getConnection();

Statement st=conn.createStatement();

String query=”select CurrentBalance from

amar˙db.sbibankcustomer where CardNumber

=””+CardNumber+””;

ResultSet rs = st.executeQuery(query);

String sql = ”update amar˙db.sbibankcustomer set

CurrentBalance = CurrentBalance - ” + Amount1 + ”where

CardNumber = ”+CardNumber;

st.execute(sql);

String query1=”select CurrentBalance from

amar˙db.hostacountdetail where AcountNo

=””+HostACNumber+””;

ResultSet rs1 = st.executeQuery(query);

String sql1 = ”update amar˙db.hostacountdetail set

CurrentBalance = CurrentBalance + ” + Amount1 +

”where

AcountNo = ”+HostACNumber;

st.execute(sql1);

SimpleDateFormat sdf = new

SimpleDateFormat(”yyyy-MM-dd”);

String date = sdf.format(new Date());

```
Calendar cal = Calendar.getInstance();
cal.getTime();
SimpleDateFormat sdf1 = new
SimpleDateFormat("HH:mm:ss");
String time = sdf1.format(cal.getTime()) ;
PreparedStatement ps = conn.prepareStatement("insert
into amar˙db.sbicusttraninfo values(?,?,?,?,?,?,?)");
String query3="select * from amar˙db.sbicusttraninfo";
st.execute(query3);
rs = st.getResultSet();
while(rs.next()){ //Retrieve by column name
Tid = rs.getString("tranID");
}
tranid=Integer.parseInt(Tid);
tranid=tranid+1;
StID=Integer.toString(tranid);
ps.setString(1, date);
ps.setString(2,time);
ps.setString(3, "Transaction Withdraw");
ps.setDouble(4, Amount1);
ps.setDouble(5, 0000);
ps.setString(6, StID);
ps.setString(7, CardNumber);
ps.executeUpdate();
ps = conn.prepareStatement("insert into
amar˙db.hostaccounttransactiondetails
values(?,?,?,?,?,?,?)");
ps.setString(1, date);
ps.setString(2, time);
ps.setString(3, "Electronic Transfer");
```

```
ps.setDouble(4, 0000);
ps.setDouble(5, Amount1);
ps.setString(6, StID);
ps.setString(7, AccountNumber);
ps.executeUpdate(); conn.commit();
rs.close(); st.close(); conn.close();
ps.close();
}catch (SQLException ex)
System.err.println(ex.getMessage());
}
TranID.value=StID;
```